Санкт-Петербургский государственный университет

Выпускная квалификационная работа аспиранта

Разработка высокопроизводительных моделей и программ динамики морских объектов

Ганкевич Иван Геннадьевич

Специальность 05.13.18
Математическое моделирование, численные методы и комплексы программ

Научный руководитель
д.т.н Дегтярев Александр Борисович

Санкт-Петербург, 2016

Содержание

Введение **4**

Постановка задачи **8**

Обзор литературы **9**

1. Моделирование ветровых волн произвольных амплитуд **14**

1.1. Основные формулы авторегрессионной модели 14

1.2. Моделирование нелинейности морских волн 16

1.3. Задача определения поля давлений 18

1.4. Двухмерное поле скоростей 20

1.5. Трехмерное поле скоростей 24

2. Постановка численного эксперимента **26**

2.1. Дополнительные формулы авторегрессионной модели 26

2.2. Сравнение полей потенциалов скоростей с известными формулами линейной теории 31

3. Распределенная объектно-ориентированная система научных расчетов **35**

3.1. Комплекс программ для многопроцессорных систем с общей памятью 37

3.2. Комплекс программ для систем с распределенной памятью 46

Заключение **63**

Выводы **64**

Список сокращений и условных обозначений **65**

Список иллюстраций 67

Список таблиц 68

Список литературы 69

Список опубликованных по теме ВКР работ 75

А. Верификация авторегрессионной модели 78

Благодарности 80

Введение

Актуальность темы. Программы, моделирующие воздействие морских волн на судно, плавучую платформу или какой-либо другой морской объект, широко используются для расчета качки судна, определения воздействия внешних сил на структуру морского объекта, а также для моделирования затопления и других процессов, вызываемых взаимодействием объекта с морскими волнами; однако, большинство из них используют линейную теорию для моделирования морского волнения [1–5], что не позволяет моделировать некоторые особенности ветроволнового климата. Среди них можно выделить переход от нормальных погодных условий к шторму и волнение, вызванное наложением множества систем ветровых волн и волн зыби, распространяющихся в разных направлениях. Другой особенностью этих программ является использование теории волн малой амплитуды, т.е. волн, амплитуда которых много меньше, чем их длина. Это делает расчеты грубыми, и не позволяет моделировать качку судна в условиях нерегулярного волнения, когда такое предположение несправедливо. Ввиду описанных сложностей разработка более совершенных моделей и методов, используемых при расчете динамики судна является актуальной задачей на сегодняшний день.

Степень разработанности. Особенностью комплексов расчета динамики судов является использование линейных моделей ветрового волнения, и, хотя эти модели хорошо исследованы, их вычислительная эффективность не всегда достаточна для проведения длительных численных экспериментов. Например, в случае модели Лонге—Хиггинса для увеличения временной протяженности реализации морского волнения может понадобится увеличить количество частот спектра на входе для исключения периодичности, что дополнительно увеличит время ее генерации. При этом, в случае альтернативных моделей ветрового волнения для расчета давлений используются методы вычислительной гидродинамики, обладающие меньшей эффективностью, чем аналитические методы, оптимизированные

для линейной модели. Таким образом, степень разработанности используемых на практике моделей ветрового волнения и методов расчета давлений не позволяет эффективно проводить длительные численные эксперименты, а уход от линейной модели приведет к качественному улучшению результатов экспериментов и будет способствовать проведению исследований возникающих редко экстремальных ситуаций, связанных с потерей остойчивости (например, опрокидывание и брочинг).

Цели и задачи. Основной целью работы является разработка математического и численного аппарата имитационного моделирования морских волн для проведения длительных численных экспериментов и определения величины создаваемого морскими волнами воздействия на морские объекты. Основные задачи, решаемые в рамках работы:

- разработка модели ветрового волнения, способной генерировать реализации взволнованной морской поверхности, имеющие сверхбольшой период и состоящие из волн произвольной амплитуды;
- разработка метода расчета давлений, работающего с этой моделью и не использующего теорию волн малой амплитуды;
- разработка комплекса программ, реализующего созданную модель и метод расчета давлений и позволяющего проводить расчеты как на многопроцессорной машине с общей памятью, так и на кластере.

Научная новизна. Авторегрессионная модель в отличие от других моделей ветрового волнения не основана на теории волн малой амплитуды, что позволяет учесть такие аспекты океанских волн, как асимметричность распределения волновых аппликат. В то же время математический аппарат этой модели хорошо разработан в других научных областях, что позволяет обобщить модель для моделирования развития морского волнения в условиях шторма с учетом климатических спектров и данных ассимиляции определенных районах мирового океана.

Теоретическая и практическая значимость работы. Применение альтернативной модели морского волнения и метода расчета давлений, не использующего предположения о малости амплитуд волн и линейной теории волн позволит качественно повысить работу комплексов программ для расчета воздействия океанских волн на морские объекты. Использование высокопроизводительных методов расчета давлений на корпус судна позволит провести более детальное исследование экстремальных ситуаций, связанных с потерей остойчивости (например, опрокидывание и брочинг), возникающих редко и требующих проведения длительных численных экспериментов.

Методология и методы исследования. Программная реализация авторегрессионной модели и методы вычислений давлений производилась сразу на нескольких языках программирования: сначала создавался прототип на инженерном языке высокого уровня (Mathematica, Octave), затем он вручную преобразовывался в программу на языке более низкого уровня. Для вывода аналитической формулы расчета давлений использовался математический аппарат, проверка осуществлялась с помощью системы компьютерной алгебры Mathematica.

Положения, выносимые на защиту.

- Модель ветрового волнения, способная генерировать реализации взволнованной морской поверхности, имеющие сверхбольшой период и состоящие из волн произвольной апмлитуды;
- Метод расчета давлений, работающий с этой моделью и не использующий теорию волн малой амплитуды;
- Комплекс программ, реализующий созданную модель и метод расчета давлений и позволяющий проводить расчеты как на многопроцессорной машине с общей памятью, так и на кластере.

Степень достоверности и апробация результатов. Достоверность полученных результатов как по модели авторегрессии так и по методу вычисле-

ний давлений подтверждается многочисленными и всесторонними численными экспериментами, целью которых было сравнение полученных результатов с поведением реальных морских волн. Апробация производилась на системе Large Amplitude Motion Program (LAMP), в которой модель авторегрессии и метод вычислений давлений были реализованы и сопоставлены с используемыми ранее методами, основанными на теории волн малой амплитуды. Эти проверки показали целесообразность применения альтернативной модели и метода расчета давлений ввиду их гидродинамической адекватности и более высокой производительности.

Постановка задачи

Задача состоит в применении авторегрессионной модели ветрового волнения для генерации морских волн произвольной амплитуды и в определении поля давлений под взволнованной морской поверхностью, сгенерированной этой моделью. Поле давлений для случая идеальной несжимаемой жидкости определяется уравнением Лапласа со смешанным граничным условием. Для случая волн малых амплитуд полученный решение должно быть сопоставимо с известными формулами линейной теории волн; для остальных случаев результат не должен расходиться. Результатом работы должна стать программная реализация авторегрессионной модели и метода вычисления давлений, эффективно работающая в распределенной вычислительной среде и готовая к включению в состав виртуального полигона.

Обзор литературы

Анализ моделей ветрового волнения

Вычисление давлений возможно только при условии знания вида взволнованной поверхности, и для ее генерации существует ряд моделей ветрового волнения, наиболее изученной из которых является модель Лонге—Хиггинса [6]. Подробный сравнительный анализ этой модели и модели авторегрессии проведен в работах [7, 8].

Модель Лонге—Хиггинса представляет взволнованную морскую поверхность в виде суперпозиции элементарных гармонических волн случайных амплитуд c_n и фаз ϵ_n, непрерывно распределенных на интервале $[0, 2\pi]$, которая определяется формулой

$$\zeta(x, y, t) = \sum_n c_n \cos(u_n x + v_n y - \omega_n t + \epsilon_n) \tag{1}$$

Волновые числа (u_n, v_n) непрерывно распределены на плоскости (u, v), т.е. площадка $du \times dv$ содержит бесконечно большое количество волновых чисел. Частота связана с волновыми числами дисперсионным соотношением $\omega_n = \omega(u_n, v_n)$, а функция $\zeta(x, y, t)$ является трехмерным эргодическим стационарным однородным гауссовым процессом, определяемым соотношением

$$2E_\zeta(u, v)\, du\, dv = \sum_n c_n^2$$

где $E_\zeta(u,v)$ — двумерная спектральная плотность энергии волн. Коэффициенты c_n определяются из энергетического спектра волнения $S(\omega)$ по формуле

$$c_n = \sqrt{\int_{\omega_n}^{\omega_{n+1}} S(\omega)d\omega}.$$

Модель Лонге—Хиггинса отличается простотой численного алгоритма и наглядностью, моделируя физически адекватную морскую поверхность, но на практике она обладает рядом недостатков, среди которых можно выделить следующие [7].

- Модель Лонге—Хиггинса рассчитана на представление стационарного гауссова поля и не подходит для решения более общих задач, что является следствием центральной предельной теоремы (сумма большого числа гармоник со случайной амплитудой и фазой будет иметь нормальное распределение в независимости от исходного распределения фаз и амплитуд).

- Модель Лонге—Хиггинса, обладая свойством периодичности, требует высокой степени дискретизации частотно направленного спектра волн, а значит и большого количества частот для проведения длительных численных экспериментов, что негативно сказывается на скорости счета.

- В численных экспериментах скорость сходимости выражения (1) низка и имеет вероятностный характер, т.к. не обеспечена сходимость по фазам ϵ_n.

- Обобщение модели для негауссовых и нелинейных процессов сопряжено с большой трудоемкостью вычислений.

Таким образом, модель Лонге—Хиггинса может быть применена для решения задач поведения динамического объекта только в линейной постановке, является неэффективной для длительных экспериментов и основана на теории волн малой амплитуды.

Известные методы определения поля давлений

Задача определения давлений под взволнованной морской поверхностью сводится к обратной задаче гидродинамики для несжимаемой невязкой жидкости. Система уравнений для несжимаемой невязкой жидкости в общем виде записывается как [9]

$$\begin{aligned} &\nabla^2\phi = 0, \\ &\phi_t + \frac{1}{2}|\vec{v}|^2 + g\zeta = -\frac{p}{\rho}, && \text{на } z = \zeta(x, y, t), \\ &D\zeta = \nabla\phi \cdot \vec{n}, && \text{на } z = \zeta(x, y, t), \end{aligned} \tag{2}$$

где ϕ — потенциал скорости, ζ — подъем (аппликата) взволнованной поверхности, p — давление жидкости, ρ — плотность жидкости, $\vec{v} = (\phi_x, \phi_y, \phi_z)$ — вектор скорости, g — ускорение свободного падения и D — субстанциональная производная (производная Лагранжа). Первое уравнение является уравнением неразрывности (уравнение Лапласа), второе — законом сохранения импульса, которое иногда называют динамическим граничным условием; третье уравнение — кинематическое граничное условие, которое сводится к равенству нормальной составляющей скорости жидкости ($\nabla\phi \cdot \vec{n}$) в каждой точке взволнованной поверхности $\zeta(x, y, t)$ скорости перемещения этой поверхности ($D\zeta$).

Обратная задача гидродинамики заключается в решении этой системы уравнений относительно ϕ. В такой постановке уравнение Лапласа и кинематическое ГУ используются для нахождения потенциала скорости, а динамическое ГУ — для вычисления давлений по известным производным потенциала. Таким образом, с математической точки зрения обратная задача гидродинамики сводится к решению уравнения Лапласа со смешанным ГУ — задаче Робена для уравнения Лапласа.

Теория волн малых амплитуд. В [10] дается решение обратной задачи гидродинамики для случая идеальной несжимаемой жидкости в рамках теории волн малых амплитуд (в предположении, что длина волны много больше ее высоты: $\lambda \gg h$). В этом случае обратная задача линейна и сводится к уравнению Лапласа со смешанным граничным условием, а уравнение движения используется только для нахождения давлений по известным значениям производных потенциала скорости. Предположение о малости амплитуд волн означает слабое изменение локального волнового числа во времени и пространстве по сравнению с подъемом (апптикатой) взволнованной поверхности. Это позволяет определить специальную формулу производной $\zeta_z = k\zeta$, где k — волновое число. Формула является основой предлагаемого решения. В двухмерном случае решение записывается как

$$\left.\frac{\partial\phi}{\partial x}\right|_{x,t} = -\frac{1}{\sqrt{1+\alpha^2}} e^{-I(x)} \int\limits_0^x \frac{\partial\dot{\zeta}/\partial z + \alpha\dot{\alpha}}{\sqrt{1+\alpha^2}} e^{I(x)} dx, \tag{3}$$

$$I(x) = \int\limits_0^x \frac{\partial\alpha/\partial z}{1+\alpha^2} dx,$$

где α — уклоны волн. В трехмерном случае решение записывается в виде эллиптического дифференциального уравнения в частных производных:

$$\frac{\partial^2\phi}{\partial x^2}\left(1+\alpha_x^2\right) + \frac{\partial^2\phi}{\partial y^2}\left(1+\alpha_y^2\right) + 2\alpha_x\alpha_y\frac{\partial^2\phi}{\partial x\partial y} +$$
$$\left(\frac{\partial\alpha_x}{\partial z} + \alpha_x\frac{\partial\alpha_x}{\partial x} + \alpha_y\frac{\partial\alpha_x}{\partial y}\right)\frac{\partial\phi}{\partial x} +$$
$$\left(\frac{\partial\alpha_y}{\partial z} + \alpha_x\frac{\partial\alpha_y}{\partial x} + \alpha_y\frac{\partial\alpha_y}{\partial y}\right)\frac{\partial\phi}{\partial y} +$$
$$\frac{\partial\dot{\zeta}}{\partial z} + \alpha_x\dot{\alpha_x} + \alpha_y\dot{\alpha_y} = 0.$$

Уравнение предполагается решаеть численно с помощью разностных методов.

Как будет показано в разд. 2.2 формула (3) расходится при попытке вычислить поле скоростей для волн больших амлитуд, а значит не может быть использована вместе с авторегрессионной моделью.

Линеаризация граничного условия. Модель Лонге—Хиггинса позволяет вывести явную формулу вычисления поля скоростей путем линеаризации кинематического граничного условия:

$$\phi(x, y, z, t) = \sum_n \frac{c_n g}{\omega_n} e^{(\sqrt{u_n^2 + v_n^2} z)} \sin(u_n x + v_n y - \omega_n t + \epsilon_n).$$

Эта формула может быть продифференцирована для получения производных потенциала и последующего вычисления давлений из динамического граничного условия.

1. Моделирование ветровых волн произвольных амплитуд

1.1. Основные формулы авторегрессионной модели

Авторегрессионная модель представляет взволнованную морскую поверхность в виде пространственно-временного поля, каждая точка которого является взвешенной суммой предыдущих по времени точек и некоторой случайной переменной (белого шума). Таким образом, состояние взволнованной поверхности в заданный момент времени находится в авторегрессионной зависимости от состояний в предыдущие моменты времени и от случайной переменной с нормальным распределением. Такая зависимость определяется соотношением

$$\zeta_{\vec{i}} = \sum_{\vec{j}=\vec{0}}^{\vec{N}} \Phi_{\vec{j}} \zeta_{\vec{i}-\vec{j}} + \epsilon_{\vec{i}},$$

где ζ — подъем (аппликата) взволнованной поверхности, Φ — коэффициенты авторегрессии, $\Phi_{\vec{0}} \equiv 0$ по определению, ϵ — белый шум, N — порядок регрессии по каждому из измерений, а стрелки обозначают многокомпонентные индексы, содержащие значение для каждого измерения. В общем случае в качестве компонент могут выступать любые скалярные величины, такие как температура, соленость и концентрация какого-либо раствора в воде.

Коэффициенты авторегрессии опеределяются из многомерных уравнений Юла—Уокера, которые получаются домножением на $\zeta_{\vec{i}-\vec{k}}$ обеих частей уравнения и взятия математического ожидания. В общем виде уравнения Юла—Уокера

записываются как

$$\gamma_{\vec{k}} = \sum_{\vec{j}=\vec{0}}^{\vec{N}} \Phi_{\vec{j}}\, \gamma_{\vec{k}-\vec{j}} + \sigma_\epsilon^2 \delta_{\vec{k}}, \qquad \delta_{\vec{k}} = \begin{cases} 1, & \text{if } \vec{k} = 0 \\ 0, & \text{if } \vec{k} \neq 0, \end{cases} \tag{4}$$

где γ — автоковариационная функция процесса ζ, σ_ϵ^2 — дисперсия белого шума. Матричная форма трехмерной системы уравнений Юла—Уокера, используемой в данной работе, имеет следующий вид.

$$\Gamma \begin{bmatrix} \Phi_{\vec{0}} \\ \Phi_{0,0,1} \\ \vdots \\ \Phi_{\vec{N}} \end{bmatrix} = \begin{bmatrix} \gamma_{0,0,0} - \sigma_\epsilon^2 \\ \gamma_{0,0,1} \\ \vdots \\ \gamma_{\vec{N}} \end{bmatrix}, \qquad \Gamma = \begin{bmatrix} \Gamma_0 & \Gamma_1 & \cdots & \Gamma_{N_1} \\ \Gamma_1 & \Gamma_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \Gamma_1 \\ \Gamma_{N_1} & \cdots & \Gamma_1 & \Gamma_0 \end{bmatrix},$$

где $\vec{N} = (N_1, N_2, N_3)$ и

$$\Gamma_i = \begin{bmatrix} \Gamma_i^0 & \Gamma_i^1 & \cdots & \Gamma_i^{N_2} \\ \Gamma_i^1 & \Gamma_i^0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \Gamma_i^1 \\ \Gamma_i^{N_2} & \cdots & \Gamma_i^1 & \Gamma_i^0 \end{bmatrix} \qquad \Gamma_i^j = \begin{bmatrix} \gamma_{i,j,0} & \gamma_{i,j,1} & \cdots & \gamma_{i,j,N_3} \\ \gamma_{i,j,1} & \gamma_{i,j,0} & \ddots & x \ \vdots \\ \vdots & \ddots & \ddots & \gamma_{i,j,1} \\ \gamma_{i,j,N_3} & \cdots & \gamma_{i,j,1} & \gamma_{i,j,0} \end{bmatrix},$$

Поскольку по определению $\Phi_{\vec{0}} \equiv 0$, то первую строку и столбец матрицы Γ можно отбросить. Матрица Γ как и оставшаяся от нее матрица будут блочно-теплицевы, положительно определены и симметричны, поэтому систему уравнений Юла—Уокера можно решить методом Холецкого, предназначенного для таких матриц.

После нахождения решения системы уравнений дисперсия белого шума определяется из уравнения (4) при $\vec{k} = \vec{0}$ как

$$\sigma_\epsilon^2 = \sigma_\zeta^2 - \sum_{\vec{j}=\vec{0}}^{\vec{N}} \Phi_{\vec{j}}\, \gamma_{\vec{j}}.$$

Гидродинамическая адекватность авторегрессионной модели. Отличительной особенностью авторегрессионной модели ветрового волнения является ее нефизическое происхождение: она возникла не в результате решения системы уравнений Навье—Стокса в некотором приближении, а как решение ряда проблем, с которыми столкнулись исследователи, использовавшие модель Лонге—Хиггинса на практике. Для использования авторегрессионной модели на практике в ряде экспериментов были исследованы различные характеристики генерируемой ей реализации и сопоставлены с соответствующими характеристиками реальной взволнованной морской поверхности.

Для авторегрессионной модели в работах [7, 8, 11] экспериментальным путем были верифицированы

- распределения различных характеристик волн (высоты волн, длины волн, длины гребней, период волн, уклон волн, показатель трехмерности),
- дисперсионное соотношение,
- сохранение интегральных характеристик для случая смешанного волнения.

Некоторые результаты экспериментов приведены в приложении А.

1.2. Моделирование нелинейности морских волн

Нелинейность морского волнения может быть учтена в рамках авторегрессионной модели путем генерации взволнованной поверхности с заданным одномерным законом распределения, что может быть достигнуто нелинейным безынерционным преобразованием случайного процесса. При этом любое нелинейное преобразование случайного процесса приводит к преобразованию его авто-

ковариационной функции, и самый простой способ подавить этот эффект состоит в предварительной трансформации автоковариационной функции процесса. Подробный метод преобразования изложен в работе [8].

Формула $z = f(y)$ преобразования взволнованной поверхности к необходимому одномерному закону распределения $F(z)$ получается путем решения нелинейного трансцендентного уравнения $F(z) = \Phi(y)$, где $\Phi(y)$ — функция одномерного нормального закона распределения. Поскольку функция распределения аппликат морских волн часто задается некоторой аппроксимацией, основанной на натурных данных, то это уравнение целесообразно решать численно в каждой точке $y_k|_{k=0}^{N}$ сетки сгенерированной поверхности относительно z_k, тогда оно запишется в виде

$$F(z_k) = \frac{1}{\sqrt{2\pi}} \int_0^{y_k} \exp\left[-\frac{t^2}{2}\right] dt, \tag{5}$$

а для его решения этого можно использовать простейший численный метод половинного деления (метод бисекции).

Для предварительного преобразования автоковариационной функции K_z процесса ее необходимо разложить в ряд по полиномам Эрмита (ряд Грама—Шарлье)

$$K_z(\vec{u}) = \sum_{m=0}^{\infty} C_m^2 \frac{K_y^m(\vec{u})}{m!},$$

где

$$C_m = \frac{1}{\sqrt{2\pi}} \int_{\infty}^{\infty} f(y) H_m(y) \exp\left[-\frac{y^2}{2}\right],$$

H_m — полином Эрмита, а $f(y)$ — решение уравнения (5). Воспользовавшись полиномиальной аппроксимацией $f(y) \approx \sum\limits_i d_i y^i$ и аналитическими выражениями для полнимов Эрмита, формулу определения коэффициентов можно упростить,

используя следующее равенство:

$$\frac{1}{\sqrt{2\pi}}\int\limits_{\infty}^{\infty} y^k \exp\left[-\frac{y^2}{2}\right] = \begin{cases} (k-1)!! & \text{для четных } k, \\ 0 & \text{для нечетных } k. \end{cases}$$

Вычисление коэффициентов C_m ведется последовательно и критерий прекращения счета определяется совпадением дисперсий обоих полей с требуемой точностью ϵ:

$$\left|\sigma_z^2 - \sum_{k=0}^{m} \frac{C_k^2}{k!}\right| \leq \epsilon.$$

В [8] автор предлагает использовать полиномиальную аппроксимацию для $f(y)$ также для преобразования поверхности, однако на практике в реализации взволнованной поверхности всегда находятся точки, выпадающие за промежуток на котором построена аппроксимация, что приводит к резкому уменьшению точности аппроксимации. В этих точках уравнение (5) эффективнее решать методом бисекции. Использование полиномиальной аппроксимацией в формулах для коэффициентов ряда Грама—Шарлье не приводит к аналогичным ошибкам.

1.3. Задача определения поля давлений

Поиск аналитических решений граничных задач для классических уравнений часто связан с исследованием различных свойств решения, и для таких исследований запись формулы общего решения неудобна ввиду своей сложности и наличия интегралов от неизвестных функций. Одним из методов нахождения аналитических решений дифференциальных уравнений в частных производных (ДУЧП) является метод Фурье. Основой метода служит преобразование Фурье,

применение которого к любому ДУЧП позволяет свести его к алгебраическому, а его решение записывается как обратное преобразование Фурье от некоторой функции (которая часто содержит преобразования Фурье от других функций). Поскольку преобразования не всегда можно записать аналитически, то вместо этого ищутся частные решения задачи и анализируется их поведение в различных областях. В то же время, вычисление дискретных преобразований Фурье на компьютере не представляет сложности ввиду наличия многочисленного семейства алгоритмов быстрого преобразования Фурье (БПФ). Эти алгоритмы используют симметрию для понижения асимптотической сложности с квадратичной $\mathcal{O}(n^2)$ до $\mathcal{O}(n \log_2 n)$. Таким образом, метод Фурье подходит для поиска аналитических частных решений ДУЧП, а общее решение, полученное этим методом, может стать основой для построения высокопроизводительных *гибридных* методов решения ДУЧП, в которых преобразования Фурье от неизвестных функций производятся численно.

Альтернативным подходом, который повсеместно применяется в решении ДУЧП, является сведение их к разностным уравнениям, решаемым с помощью построения различных численных схем. При этом решение получается приближенным, а сложность алгоритмов сопоставима со сложностью алгоритма БПФ. Например, стационарное эллиптическое уравнение в частных производных преобразуется в неявную разностную схему, решаемую итерационным методом, на каждом шаге которого ищется решение трехдиагональной или пятидиагональной СЛАУ методом прогонки (алгоритм Томаса). Асимптотическая сложность алгоритма составляет $\mathcal{O}(nm)$, где n — количество точек на сетке взволнованной поверхности, m — число итераций.

С вычислительной точки зрения наличие большого количества преобразований Фурье в решении является преимуществом. Решения полученные с помощью метода Фурье явные, а значит хорошо масштабируются на большое количество параллельно работающих вычислительных ядер с использованием простейших приемов параллельного программирования. Эти преимущества обусловили вы-

бор метода Фурье в качестве рабочего для получения явного аналитического решения задачи определения давлений под взволнованной морской поверхностью.

1.4. Двухмерное поле скоростей

Формула для жидкости бесконечной глубины. Задача Робена для уравнения Лапласа в двух измерениях записывается как

$$\begin{aligned} &\phi_{xx} + \phi_{zz} = 0, && (6)\\ &\zeta_t + \zeta_x \phi_x = \frac{\zeta_x}{\sqrt{1+\zeta_x^2}}\phi_x - \phi_z, && \text{на } z = \zeta(x,t). \end{aligned}$$

Для ее решения воспользуемся методом Фурье. Возьмем преобразование Фурье от обоих частей уравнений Лапласа и получим

$$-4\pi^2 \left(u^2 + v^2\right) \mathcal{F}\left\{\phi(x,z)\right\}(u,v) = 0,$$

откуда имеем $v = \pm iu$. Здесь и далее будет использоваться следующая симметричная форма преобразования Фурье:

$$\mathcal{F}\left\{f(x,y)\right\}(u,v) = \iint\limits_{-\infty}^{\infty} f(x,y) e^{-2\pi i(xu+yv)} dxdy$$

Решение уравнения будем искать в виде обратного преобразования Фурье $\phi(x,z) = \mathcal{F}^{-1}\left\{E(u,v)\right\}(x,z)$. Применяя полученное равенство со знаком «+»[1], решение

[1] Выражение $v = -iu$ не подходит в данной задаче, поскольку потенциал скорости должен стремиться к нулю с увеличением глубины (с уменьшением z).

можно переписать как

$$\phi(x,z) = \mathcal{F}^{-1}\left\{e^{2\pi uz}E(u)\right\}(x). \tag{7}$$

Для того чтобы подстановка $z = \zeta(x,t)$ не помешала использованию преобразований Фурье в решении, перепишем (7) в виде свертки:

$$\phi(x,z) = \mathcal{D}_1(x,z) * \mathcal{F}^{-1}\left\{E(u)\right\}(x),$$

где $\mathcal{D}_1(x,z)$ — некоторая функция, вид которой будет определен в разд. 2.2 и для которой выполняется соотношение $\mathcal{F}\left\{\mathcal{D}_1(x,z)\right\}(u) = e^{2\pi uz}$. Подставляя выражение для ϕ в граничное условие, получим

$$\zeta_t = (if(x) - 1)\left[\mathcal{D}_1(x,z) * \mathcal{F}^{-1}\left\{2\pi uE(u)\right\}(x)\right],$$

где $f(x) = \zeta_x/\sqrt{1+\zeta_x^2} - \zeta_x$. Применяя преобразование Фурье к обеим частям, получаем выражение для коэффициентов E:

$$E(u) = \frac{1}{2\pi u}\frac{\mathcal{F}\left\{\zeta_t/\left(if(x)-1\right)\right\}(u)}{\mathcal{F}\left\{\mathcal{D}_1(x,z)\right\}(u)}$$

Выполняя подстановку $z = \zeta(x,t)$ и подставляя полученное выражение в (7), получаем окончательное выражение для $\phi(x,z)$:

$$\boxed{\phi(x,z) = \mathcal{F}^{-1}\left\{\frac{e^{2\pi uz}}{2\pi u}\frac{\mathcal{F}\left\{\zeta_t/\left(if(x)-1\right)\right\}(u)}{\mathcal{F}\left\{\mathcal{D}_1(x,\zeta(x,t))\right\}(u)}\right\}(x).} \tag{8}$$

Множитель $e^{2\pi uz}/(2\pi u)$ делает график функции от которой берется обратное преобразования Фурье несимметричным относительно оси OY. Это затрудняет применение БПФ, поскольку оно требует периодичную функцию, которая на концах промежутка принимает нулевое значение. Реализация же этого обратного преобразования с помощью численного интегрирования не позволит получить

преимущество над решением всей системы уравнений с помощью разностных схем. Решением проблемы является использование формулы (10) для жидкости конечной глубины с заведомо большим значением глубины водоема h.

Формула для жидкости конечной глубины. На дне водоема вертикальная составляющая скорости перемещения жидкости должна равняться нулю, т.е. $\phi_z = 0$ на $z = -h$, где h — глубина водоема. В этом случае пренебречь равенством $v = -iu$, полученным из уравнения Лапласа, нельзя, и решение ищется в виде

$$\phi(x,z) = \mathcal{F}^{-1}\left\{\left(C_1 e^{2\pi u z} + C_2 e^{-2\pi u z}\right) E(u)\right\}(x). \tag{9}$$

Подставляя ϕ в условие на дне водоема, получим

$$C_1 e^{-2\pi u h} - C_2 e^{2\pi u h} = 0,$$

откуда имеем $C_1 = \frac{1}{2} C e^{2\pi u h}$ и $C_2 = -\frac{1}{2} C e^{-2\pi u h}$. Константа C здесь произвольна, поскольку при подстановке станет частью неизвестных коэффициентов $E(u)$. Подставляя полученные выражения для C_1 и C_2 в (9), получаем выражение

$$\phi(x,z) = \mathcal{F}^{-1}\left\{\cosh\left(2\pi u(z+h)\right) E(u)\right\}(x).$$

Выполняя аналогичные предыдущему разделу операции, получаем окончательное выражение для $\phi(x,z)$:

$$\boxed{\phi(x,z,t) = \mathcal{F}^{-1}\left\{\frac{\cosh\left(2\pi u(z+h)\right)}{2\pi u}\frac{\mathcal{F}\left\{\zeta_t / \left(if(x)-1\right)\right\}(u)}{\mathcal{F}\left\{\mathcal{D}_2\left(x,\zeta(x,t)\right)\right\}(u)}\right\}(x),} \tag{10}$$

где $\mathcal{D}_2(x,z)$ — некоторая функция, вид которой будет определен в разд. 2.2 и для которой выполняется соотношение $\mathcal{F}\left\{\mathcal{D}_2(x,z)\right\}(u) = \cosh(2\pi u z)$.

Сведение к формулам линейной теории волн. Справедливость полученных формул проверим, подставив в качестве $\zeta(x,t)$ известные аналитические выражения для плоских волн. Для вычисления преобразований Фурье символьно

воспользуемся пакетом Mathematica [12]. В линейной теории широко используется предположение о малости амплитуд волн, что позволяет упростить исходную систему уравнений (6) до

$$\begin{aligned} &\phi_{xx} + \phi_{zz} = 0, \\ &\zeta_t = -\phi_z \qquad\qquad\qquad\qquad \text{на } z = \zeta(x,t), \end{aligned}$$

решение которой запишется как

$$\phi(x,z,t) = -\mathcal{F}^{-1}\left\{\frac{e^{2\pi u z}}{2\pi u}\mathcal{F}\left\{\zeta_t\right\}(u)\right\}(x).$$

Профиль прогрессивной волны описывается формулой $\zeta(x,t) = A\cos(2\pi(kx - t))$. Подстановка этого выражения в формулу для бесконечной глубины дает равенство $\phi(x,z,t) = -\frac{A}{k}\sin(2\pi(kx-t))\cosh(2\pi kz)$. Чтобы свести его к формуле линейной теории волн, необходимо представить гиперболический синус в экспоненциальной форме и отбросить член, содержащий $e^{-2\pi kz}$, как противоречащий условию $\phi \underset{z\to-\infty}{\longrightarrow} 0$. После взятия действительной части выражения получится формула линейной теории $\phi(x,z,t) = \frac{A}{k}e^{2\pi kz}\sin(2\pi(kx-t))$. Аналогично, предположение о малости амплитуд волн позволяет упростить формулу (10) до

$$\phi(x,z,t) = -\mathcal{F}^{-1}\left\{\frac{\cosh(2\pi u(z+h))}{2\pi u\cosh(2\pi uh)}\mathcal{F}\left\{\zeta_t\right\}(u)\right\}(x).$$

Подстановка формулы для прогрессивной плоской волны в это выражение дает равенство

$$\phi(x,z,t) = \frac{A}{k}\frac{\cosh(2\pi k(z+h))}{\cosh(2\pi kh)}\sin(2\pi(kx-t)), \tag{11}$$

что соответствует формуле линейной теории для конечной глубины.

Различная запись решения уравнения Лапласа, в котором затухающая экспонента может встречаться как со знаком «+», так и со знаком «-», может стать причиной разницы в формуле, где вместо cosh будет использоваться sinh. Выра-

жение $\frac{\cosh(2\pi k(z+h))}{\cosh(2\pi kh)} \approx \frac{\cosh(2\pi k(z+h))}{\cosh(2\pi kh)}$ превращается в строгое равенство на поверхности, и разница между правой левой частью увеличивается при приближении к дну водоема (для достаточно большой глубины ошибка вблизи поверхности жидкости незначительна), поэтому на практике для расчетов вблизи взволнованной поверности можно использовать любую из функций.

Сведение формул (8) и (10) к формулам линейной теории волн показало, что формула (8) не подходит для расчета потенциала на бесконечной глубине с использованием численных методов, т.к. не обладает необходимой для применения быстрого преобразования Фурье симметрией. Для такого случая можно использовать формулу для конечной глубины, полагая h равным характерному значению глубины исследуемого водоема. Для стоячих волн сведение к формулам линейной теории происходит с аналогичными предположениями.

1.5. Трехмерное поле скоростей

В трех измерениях исходная система уравнений (2) переписывается как

$$\phi_x x + \phi_y y + \phi_z z = 0, \tag{12}$$
$$\zeta_t + \zeta_x\phi_x + \zeta_y\phi_y = \frac{\zeta_x}{\sqrt{1+\zeta_x^2}}\phi_x + \frac{\zeta_y}{\sqrt{1+\zeta_y^2}}\phi_y - \phi_z, \qquad \text{на } z = \zeta(x,y,t).$$

Для ее решения воспользуемся методом Фурье. Возьмем преобразование Фурье от обоих частей уравнений Лапласа и получим

$$-4\pi^2\left(u^2+v^2+w^2\right)\mathcal{F}\left\{\phi(x,y,z)\right\}(u,v,w) = 0,$$

откуда имеем $w = \pm i\sqrt{u^2+v^2}$. Решение уравнения будем искать в виде обратного преобразования Фурье $\phi(x,y,z) = \mathcal{F}^{-1}\left\{E(u,v,w)\right\}(x,y,z)$. Применяя полученное равенство, получаем

$$\phi(x,y,z) = \mathcal{F}^{-1}\left\{\left(C_1 e^{2\pi\sqrt{u^2+v^2}z} - C_2 e^{-2\pi\sqrt{u^2+v^2}z}\right) E(u,v)\right\}(x,y).$$

Подставляя ϕ в условие на дне водоема аналогично двухмерному случаю, получаем

$$\phi(x,y,z) = \mathcal{F}^{-1}\left\{\cosh\left(2\pi\sqrt{u^2+v^2}(z+h)\right) E(u,v)\right\}(x,y). \tag{13}$$

Подставляя выражение для ϕ в граничное условие, получим

$$\begin{aligned}\zeta_t = {} & i f_1(x)\mathcal{F}^{-1}\left\{2\pi u \cosh\left(2\pi\sqrt{u^2+v^2}(z+h)\right) E(u,v)\right\}(x,y) \\ & + i f_2(x)\mathcal{F}^{-1}\left\{2\pi v \cosh\left(2\pi\sqrt{u^2+v^2}(z+h)\right) E(u,v)\right\}(x,y) \\ & - \mathcal{F}^{-1}\left\{2\pi\sqrt{u^2+v^2}\cosh\left(2\pi\sqrt{u^2+v^2}(z+h)\right) E(u,v)\right\}(x,y)\end{aligned}$$

где $f_1(x,y) = \zeta_x/\sqrt{1+\zeta_x^2} - \zeta_x$ и $f_2(x,y) = \zeta_y/\sqrt{1+\zeta_y^2} - \zeta_y$. Применяя преобразование Фурье к обеим частям, получаем выражение для коэффициентов E:

$$\begin{aligned}\mathcal{F}\left\{\zeta_t\right\}(u,v) = {} & \mathcal{F}\left\{i f_1(x)\mathcal{F}^{-1}\left\{2\pi u \cosh\left(2\pi\sqrt{u^2+v^2}(z+h)\right)\right\}(x,y)\right\}(u,v) E(u,v) \\ & + \mathcal{F}\left\{i f_2(x)\mathcal{F}^{-1}\left\{2\pi v \cosh\left(2\pi\sqrt{u^2+v^2}(z+h)\right)\right\}(x,y)\right\}(u,v) E(u,v) \\ & - 2\pi\sqrt{u^2+v^2}\cosh\left(2\pi\sqrt{u^2+v^2}(z+h)\right) E(u,v)\end{aligned}$$

Окончательное решение получается при подстановке выражения для $E(u,v)$ в (13).

2. Постановка численного эксперимента

2.1. Дополнительные формулы авторегрессионной модели

Входными параметрами генератора взволвнованной морской поверхности служат автоковариацонная функция и фнукция распределения волновых аппликат. Обе функции можно задать полиномиальной аппроксимацией натурных данных или аналитически.

Аппроксимация АКФ. Аналитические выражения для АКФ получаются из соответствующих выражений для профиля морской волны путем численного интегрирования и последующей аппроксимации модельной функцией. В качестве выражений для профиля морской волны берутся известные из линейной теории формулы, домноженные на затухающую экспоненту. Вычисление АКФ в таком случае можно провести аналитически с помощью систем компьютерной алгебры, используя теорему Винера—Хинчина: $K(u,v) = \mathcal{F}\left\{\zeta(x,t)^2\right\}(u,v)$, где K — АКФ, ζ — профиль морской волны, $\mathcal{F}$ — преобразование Фурье. Получившиеся после взятия преобразования Фурье выражения аппроксимируются модельной АКФ по формуле

$$K(u,v) = \gamma \exp(-\alpha(|u| + |v|)) \cos(\beta(u+v)).$$

Значения коэффициентов α, β, γ для прогрессивных и стоячих волн показаны в табл. 1.

Аппроксимация распределения аппликат. В [13] было экспериментально показано, что распределение аппликат взволнованной морской поверхности

Профиль волны	Весовая функция	Значения коэффициентов
$A\cos(2\pi(kx-\omega t))$	$\exp(-(\lvert t\rvert+\lvert x\rvert))$	$\alpha = 0.394279$ $\beta = 0.885028$ $\gamma = 0.0106085$
$A\sin(kx)\sin(\sigma t)$	$\exp(-(\lvert t\rvert+\lvert x\rvert))$	$\alpha = 0.352495$ $\beta = -0.0874116$ $\gamma = 0.0851927$

Таблица 1: Значения параметров аппроксимации АКФ (2.1) для прогрессивных и стоячих волн ($A = 1, k = 1, \omega = 1, \sigma = 1$).

отличается от нормального ненулевым экцессом и асимметрией. В [14] показано, что такое распределение можно разложить в ряд Грама—Шарлье:

$$\begin{aligned} F(z;\gamma_1,\gamma_2) &= \phi(z) - \gamma_1\frac{\phi'''(z)}{3!} + \gamma_2\frac{\phi''''(z)}{4!} \\ &= \frac{1}{2}\text{erf}\left[\frac{z}{\sqrt{2}}\right] - \frac{e^{-\frac{z^2}{2}}}{\sqrt{2\pi}}\left[\frac{1}{6}\gamma_1\left(z^2-1\right) + \frac{1}{24}\gamma_2 z\left(z^2-3\right)\right], \\ f(z;\gamma_1,\gamma_2) &= \frac{e^{-\frac{z^2}{2}}}{\sqrt{2\pi}}\left[\frac{1}{6}\gamma_1 z\left(z^2-3\right) + \frac{1}{24}\gamma_2\left(z^4-6z^2+3\right)+1\right], \end{aligned} \tag{14}$$

где $\phi(z) = \frac{1}{2}\text{erf}(z/\sqrt{2})$, γ_1 — асимметрия, γ_2 — эксцесс, f — плотность распределения, F — функция распределения. Согласно [15] для аппликат морских волн значение асимметрии выбирается на интервале $0.1 \leq \gamma_1 \leq 0,52]$, а значение эксцесса на интервале $0,1 \leq \gamma_2 \leq 0,7$. Вид плотности распределения при различных параметрах показан на рис. 1.

Альтернативной аппроксимацией распределения волновых аппликат служит формула асимметричного нормального распределения:

$$\begin{aligned} F(z;\alpha) &= \frac{1}{2}\text{erfc}\left[-\frac{z}{\sqrt{2}}\right] - 2T(z,\alpha), \\ f(z;\alpha) &= \frac{e^{-\frac{z^2}{2}}}{\sqrt{2\pi}}\text{erfc}\left[-\frac{\alpha z}{\sqrt{2}}\right], \end{aligned} \tag{15}$$

где T — функция Оуэна [16].

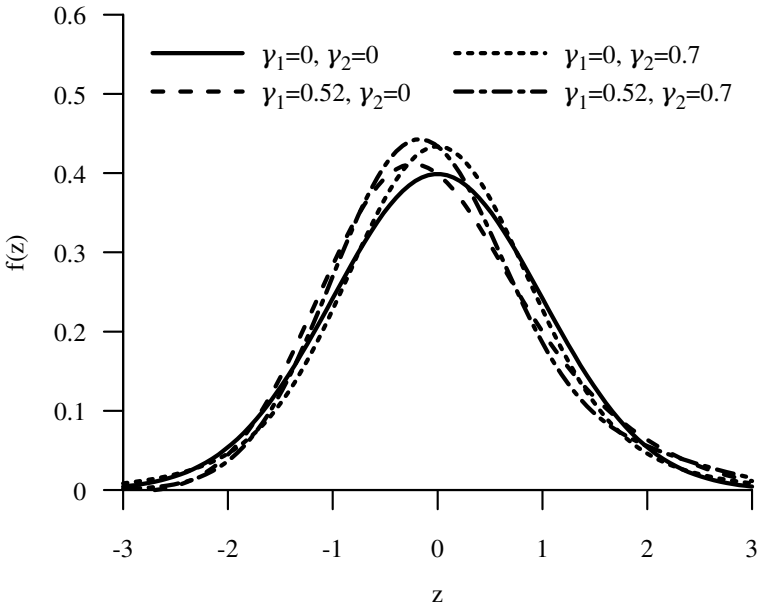

Рис. 1: Вид плотности распределения (14) волновых аппликат при различных значениях асимметрии γ_1 и эксцесса γ_2.

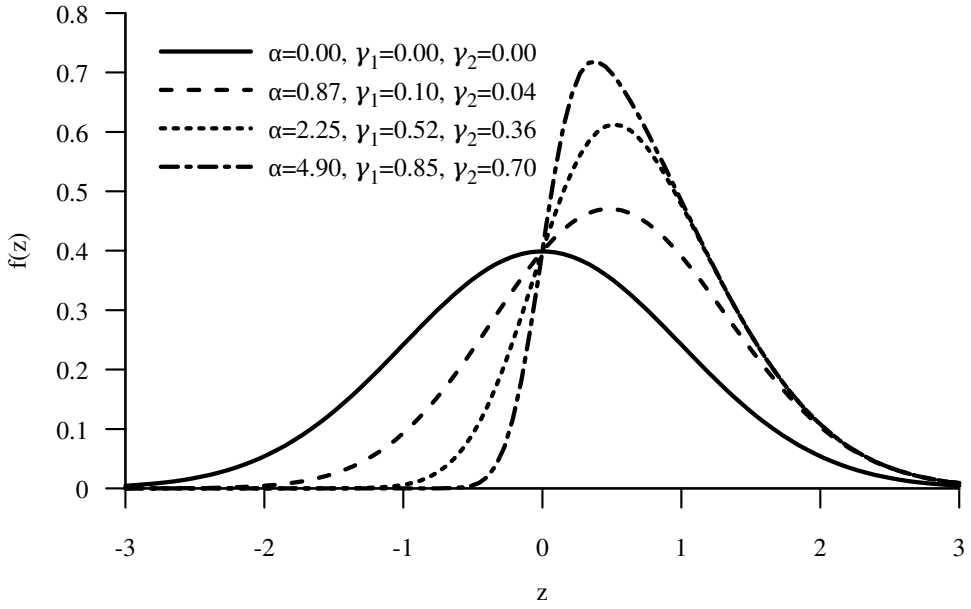

Рис. 2: Вид плотности распределения (15) волновых аппликат при различных значениях коэффициента асимметрии α.

Решение уравнения (5) с выбранной функцией распределения можно произвести в каждой точке поверхности, что даст более точные результаты, но с вычислительной точки зрения эффективнее решить это уравнение в фиксированных узлах, а затем интерполировать решение методом наименьших квадратов (МНК). Функция аппроксимировалась многочленом 12 порядка на интерполяционной сетке из 500 узлов на промежутке $-5\sigma_z \leq z \leq 5\sigma_z$. Увеличение порядка многочлена приводило либо к переполнениям при интерполяции МНК, либо к дополнительным коэффициентам близким к нулю; увеличение количества узлов также незначительно влияло на результат. В большинстве случаев трех коэффициентов ряда Грама—Шарлье было достаточно для преобразования автоковариационной функции; интерполяция увеличила относительную погрешность с 10^{-5} до $0,43 \cdot 10^{-3}$.

Генерация белого шума. Чтобы исключить периодичность из сгенерированной моделью ветрового волнения реализации взволнованной поверхности, нужно использовать генератор псевдослучайных чисел (ГПСЧ) с достаточно большим периодом. В качестве такого генератора в работе используется параллельная реализация вихря Мерсенна [17] с периодом $2^{19937} - 1$. Если предположить, что взволнованная поверхность генерируется на сетке размера 1000×1000 и шаг по времени составляет 1 сек., то вихрь Мерсенна позволяет сгенерировать неповторяющуюся последовательность астрономической временной протяженности. Этого более чем достаточно для практического применения генератора в задаче моделирования морского волнения.

Запуск нескольких ГПСЧ с разными начальными состояниями в параллельных потоках не гарантирует некоррелированность генерируемых последовательностей псевдослучайных чисел, и, чтобы избежать этого, в работе используется алгоритм динамического создания вихрей Мерсенна [18]. Суть алгоритма заключается в поиске таких матриц начальных состояний генераторов, которые бы дали максимально некоррелированные последовательности псевдослучайных чисел при параллельном запуске нескольких вихрей Мерсенна с этими начальными состоянями. Поскольку поиск начальных состояний может занимать значительно

большее время чем генерация белого шума, то вектор состояний создается предварительно для заведомо большего количества параллельных потоков. Вектор состояний сохраняется в файл, который впоследствиии считывается основной программой перед началом генерации псевдослучаных чисел.

Генерация взволнованной поверхности. В авторегрессионной модели каждая значение подъема взволнованной поверхности в каждой точке зависит от предыдующих по пространству и времени значений, из-за чего в начале реализации образуется *интервал разгона* — промежуток, на котором реализация не соответствуют заданной АКФ. Способ решения этой проблемы зависит от контекста, в котором используется реализация.

- Если реализация используется в контексте расчета остойчивости судна без учета маневрирования, то интервал никак не повлияет результаты эксперимента, поскольку находится на границе (далеко от исследуемого морского объекта).

- Если изучается остойчивость судна в условиях маневрирования, то интервал проще всего исключить из реализации (размер интервала примерно равен числу коэффициентов авторегрессии по каждому из измерений).

- Альтернативным подходом является предварительная генерация взволнованной поверхности на интервале разгона моделью Лонге—Хиггинса и генерация остальной реализации с помощью авторегрессионной модели.

В параллельном алгоритме генерации взволнованной поверхности используется параллелизм по данным: реализация делится на равные части, каждая из которых генерируется независимо, — однако, в начале каждой из частей также присутствует интервал разгона. Для его исключения подходит метод *сшивания*, часто применяемый в обработке цифровых сигналов [19–21]. Суть метода заключается в добавлении интервала равного по размеру интервалу разгона в конец каждой из частей. На этом интервале также генерируется взволнованная поверхность,

Функция	Без нормировки	С нормировкой
$\mathcal{D}_1(x,z)$	$\delta(x+iz)$	$\frac{1}{2h}\text{sech}\left(\frac{\pi(x-i(h+z))}{2h}\right)$
$\mathcal{D}_2(x,z)$	$\frac{1}{2}\left[\delta(x-iz)+\delta(x+iz)\right]$	$\frac{1}{4h}\left[\text{sech}\left(\frac{\pi(x-i(h+z))}{2h}\right)+\text{sech}\left(\frac{\pi(x+i(h+z))}{2h}\right)\right]$

Таблица 2: Формулы вычисления функций $\mathcal{D}_1(x,z)$ и $\mathcal{D}_2(x,z)$ из разд. 1.4, использующие нормировку для исключения неоднозначности определения дельта функции комплексного аргумента.

а после генерации интервал в конце части N накладывается на интервал разгона в начале части $N+1$, и значения в соответсвтующих точках складываются.

2.2. Сравнение полей потенциалов скоростей с известными формулами линейной теории

В решениях двухмерной задачи (8) и (10) присутствуют функции $\mathcal{D}_1(x,z) = \mathcal{F}^{-1}\left\{e^{2\pi uz}\right\}(x)$ и $\mathcal{D}_2(x,z) = \mathcal{F}^{-1}\left\{\cosh(2\pi uz)\right\}(x)$, которые могут быть представлены аналитически различными выражениями, представляющими сложность для вычислений на компьютере. Каждая из функций является преобразованием Фурье от линейной комбинации экспонент, которое для таких функций определено неоднозначно (см. 2). Для получения однозначного аналитического выражения можно воспользоваться нормировкой $1/\cosh(2\pi uh)$, которая также должна быть включена в выражение для коэффициентов $E(u)$. Численные эксперименты показали, что нормировка хоть и позволяет получить решение с адекватными величинами потенциалов скорости, но оно мало отличается от выражений из линейной теории волн, в которых члены с ζ опускаются.

Сравнение полученных общих формул (8) и (10) с известными формулами линейной теории волн позволяет оценить различие между полями скоростей

для волн как больших, так и малых амплитуд. В общем случае получить аналитическое выражение даже для плоских волн не представляется возможным, поэтому сравнение производится численно. Имея ввиду выводы предыдущего раздела, сравниваются только формулы для случая конечной глубины.

Отличие от формул линейной теории. В ходе численного эксперимента результаты, полученные по формуле (10) для конечной глубины были сопоставлены с результатами, полученными по соответсвующей формуле (11) линейной теории, и проверка показала качественные различия в получившихся полях потенциалов скоростей (см. рис. 3). Во-первых, потенциальные линии имеют вид затухающей синусоиды, что отличается от овальной формы, описываемой в линейой теории волн. Во-вторых, по мере приближения к дну водоема потенциал гораздо быстрее затухает, чем описывается в линейной теории, а область, где сконцентрирована большая часть энергии волны, еще больше приближена к ее гребню. Аналогичный численный эксперимент, в котором из формулы (10) были исключены члены, которыми пренебрегают в рамках линейной теории волн, показал, что полное соотвествие получившихся полей потенциалов скоростей (насколько это позволяет сделать машинная точность).

Отличие от формул теории волн малой амплитуды. В ходе численного эксперимента результаты, полученные по формуле (10) были сопоставлены с результатами, полученными по формуле для волн малой амплитуды (3), и проверка показала схожесть полей скоростей, вычисленных этими методами. Для определения скоростей использовалась реализации морской поверхности, построенные по авторегрессионной модели и различающиеся амплитудой волн. При этом интегрирование в формуле (10) велось по соответствующему сгенерированной морской поверхности интервал волновых чисел. Эксперименты проводились для волн разных амплитуд, и для волн малой амплитуды оба метода показывают сопоставимые результаты, в то время как для волн высоких амплитуд стабильное поле скоростей дает только формула (10) (см. рис. 4). Таким образом, получен-

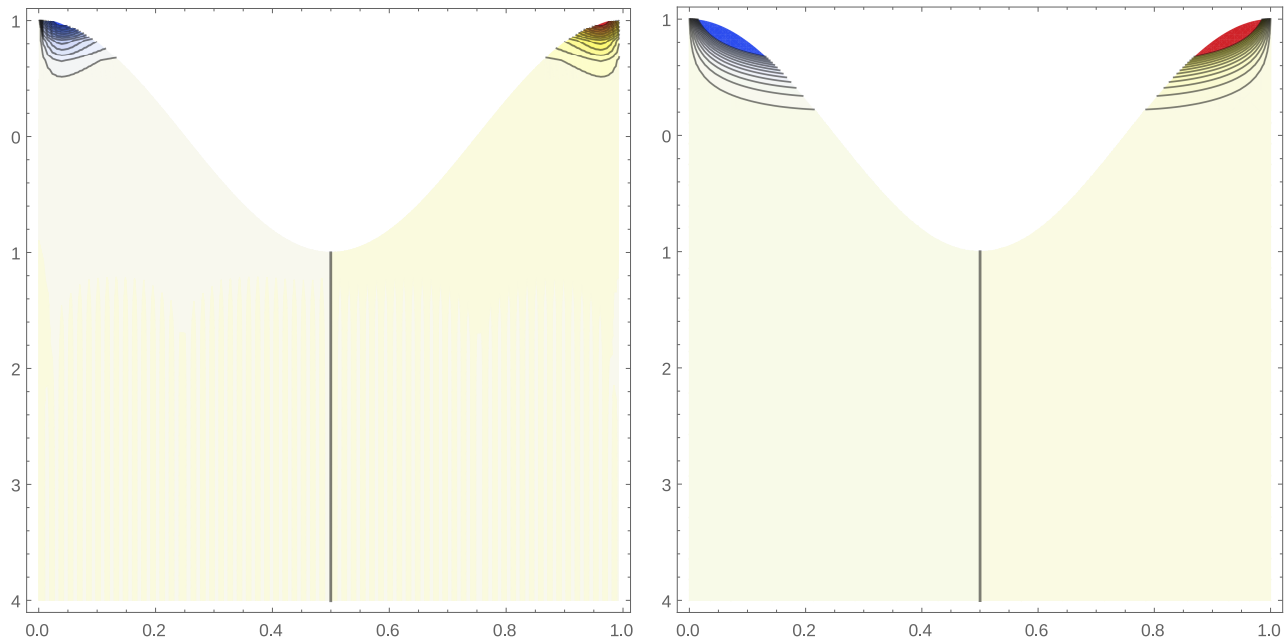

Рис. 3: Поле потенциала скорости прогрессивной волны $\zeta(x, y, t) = \cos(2\pi x - t/2)$. Поле, полученное по формуле (10) (слева) и по формуле линейной теории волн (справа).

ная формула показывает удовлетворительные результаты для различных морских поверхностей, не вводя ограничения на величины амплитуд волн.

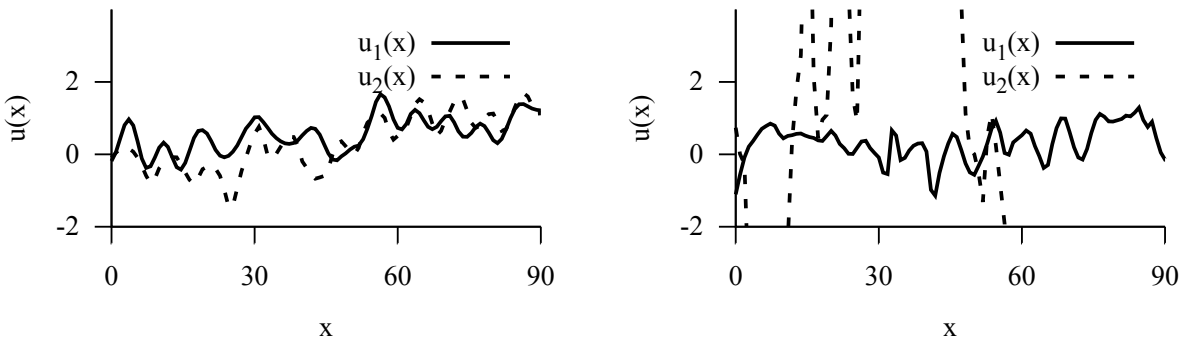

Рис. 4: Сравнение полей скоростей на поверхности моря, полученных по общей формуле (u_1) и формуле для волн малой амплитуды (u_2). Поле скоростей для поверхности волн малой амплитуды (слева) и большой амплитуды (справа).

3. Распределенная объектно-ориентированная система научных расчетов

Авторегрессионная модель ветрового волнения реализована в программном комплексе, работающем по принципу вычислительного конвейера, в котором каждое звено применяет линейный фильтр к выходным данным предыдущего звена. Такая архитектура позволяет распределить различные звенья конвейера по узлам вычислительного кластера, т.е. реализовать параллелизм по операциям, а внутри каждого звена применить параллелизм по данным. На рис. 5 представлена UML-диаграмма деятельности конвейера обработки данных, в которой прямоугольниками со скругленными углами обозначены звенья конвейера, обычными прямоугольниками — массивы данных, передаваемые от одного звена к другому, а стрелками — направление передачи данных. Некоторые звенья разделены на *секции*, каждая из которых обрабатывает разные объекты предметной области задачи. Если хотя бы две секции одного звена соединены стрелками с двумя секциями другого звена, то передача объектов между такими звеньями происходит параллельно с вычислениями, по мере их готовности. Секции работают параллельно на нескольких ядрах процессора (нескольких узлах кластера). Таким образом, между множеством ядер процессора, секций конвейера и объектами устанавливается сюръективное отображение, т.е. на одном ядре процессора может работать несколько секций конвейера, каждая из которых может обрабатывать несколько частей данных последовательно, но одна секция не может работать сразу на нескольких ядрах, а объект не может обрабаываться сразу несколькими секциями конвейера.

Модель конвейера объектов можно считать развитием модели BSP (Bulk synchronous parallel), широко применяемой в системах обработки графов [22, 23].

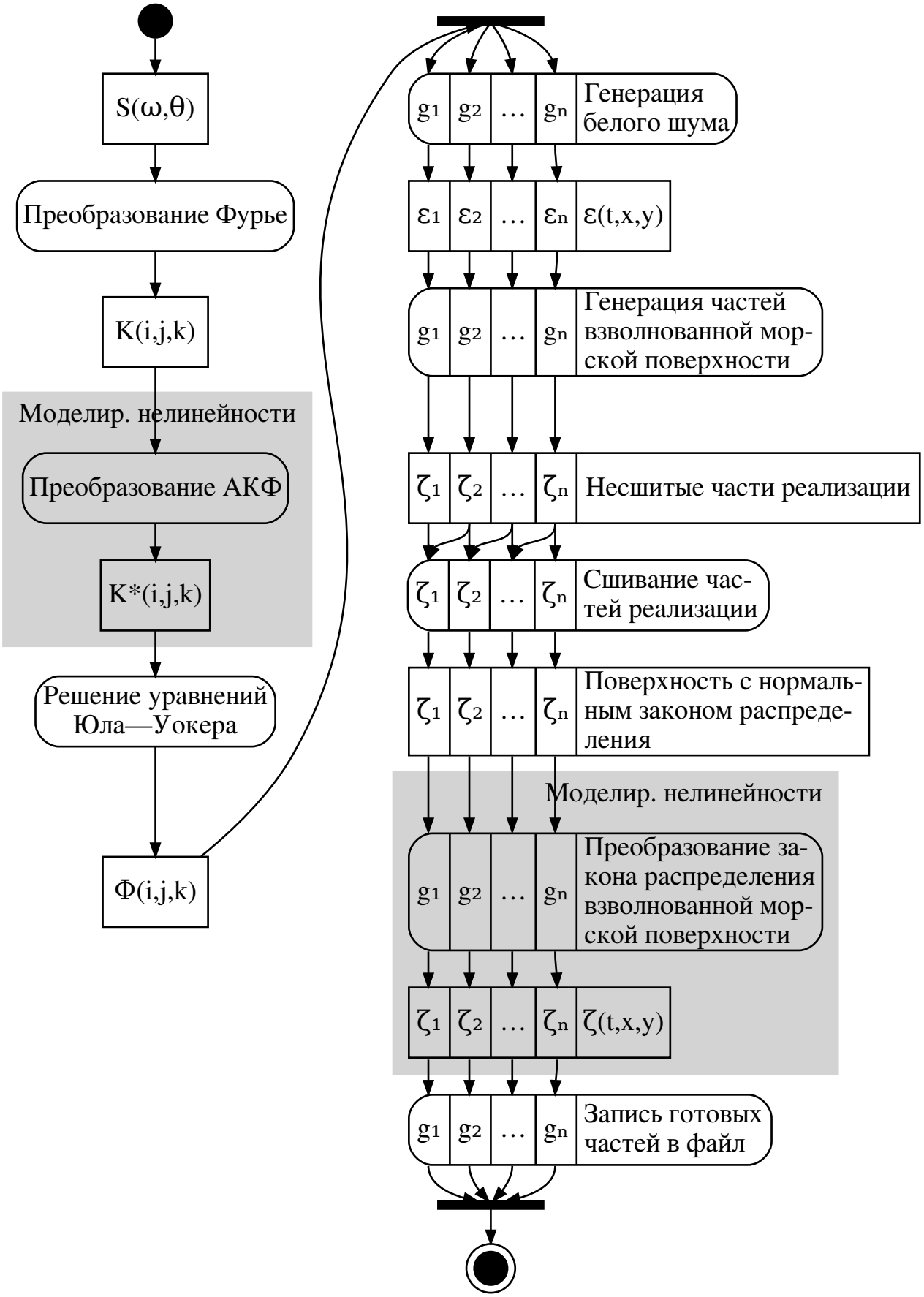

Рис. 5: Схема конвейера обработки данных, реализующего генерацию взволнованной морской поверхности по АР модели.

Преимущество конвейера заключается в том, что передача данных между звеньями, происходящая параллельно с вычислениями, позволяет исключить глобальную синхронизацию между последовательно идущим этапами вычислений, что, в свою очередь, позволяет эффективно распределить объекты по узлам кластера. В то же время, в модели BSP глобальная синхронизация происходит после каждого шага.

3.1. Комплекс программ для многопроцессорных систем с общей памятью

Программная реализация состояла в создании и отладке прототипа программы и в последующем написании компоненты виртуального полигона на языке более низкого уровня. При этом тесты показали, что одной высокопроизводительной многопроцессорной машины достаточно для создания типовых реализаций морского волнения. Также использование видеокарт в качестве векторных ускорителей эффективно только в случае расчета давлений, в то время как генерация волновой поверхности выполняется быстрее на скалярном процессоре [24].

Создание программной реализации происходило в два этапа: на первом этапе был создан и отлажен прототип в программной среде Mathematica [12], а на втором этапе логика программы была переписана на более низкоуровневом языке C++, и для получения эффективно работающего параллельного кода были проведены эксперименты с рядом библиотек. С помощью этих библиотек были реализованы функции генерации взволнованной морской поверхности, а также процедура расчета гидродинамических давлений под сгенерированной поверхностью. Тестирование производилось на вычислительных машинах кластера РЦ ВЦ СПбГУ (см. табл. 3) и позволило получить два основных результата. Во-первых, ис-

Вычислительная машина	HP SL390s G7
Процессор	2 × Intel X5650 (всего 12 ядер)
Оперативная память	96ГБ RAM
Операционная система	CentOS 5.6 (Linux)

Таблица 3: Конфигурация оборудования.

	Модель Лонге—Хиггинса			Авторег. модель		
Размер	OpenCL	OpenMP	MPI	OpenCL	OpenMP	MPI
400000	0.82	40.44	32.60	1.80	0.800	0.750
440000	0.90	44.59	35.78	1.92	0.100	0.930
480000	0.99	48.49	38.93	2.29	0.970	0.126
520000	1.07	52.65	41.92	2.43	0.118	0.117
560000	1.15	56.45	45.00	2.51	0.117	0.161
600000	1.23	60.85	48.80	2.54	0.123	0.132
640000	1.31	65.07	53.02	2.73	0.123	0.160
680000	1.40	68.90	54.92	2.80	0.138	0.136
720000	1.48	72.49	58.42	2.88	0.144	0.173
760000	1.56	76.86	61.41	3.47	0.156	0.155
800000	1.64	81.03	66.42	3.25	0.166	0.174

Таблица 4: Время (с.) генерации взволнованной морской поверхности различными программными реализациями авторегрессионной модели.

пользование видеокарт неэффективно при генерации волновой поверхности (см. табл. 4 что обусловлено сравнительно небольшим количеством арифметических операций по отношению к количеству операций с памятью устройства, а также отсутствием трансцендентных функций в реализации алгоритма [24]. Во вторых, для генерации одной реализации взволнованной морского поверхности одной многопроцессорной машины достаточно для эффективного и быстрого решения задачи (см. рис. 6). По результатам тестирования стандарт OpenMP был выбран в качестве основного, как наиболее эффективный и наиболее подходящий для расчетов на многопроцессорной системе. Кроме выбора стандарта параллельных вычислений на время работы программы влияет выбор библиотек типовых вычислительных методов, и эффективность этих библиотек была показана тестированием их разработчиками. В качестве библиотеки для матричных операций

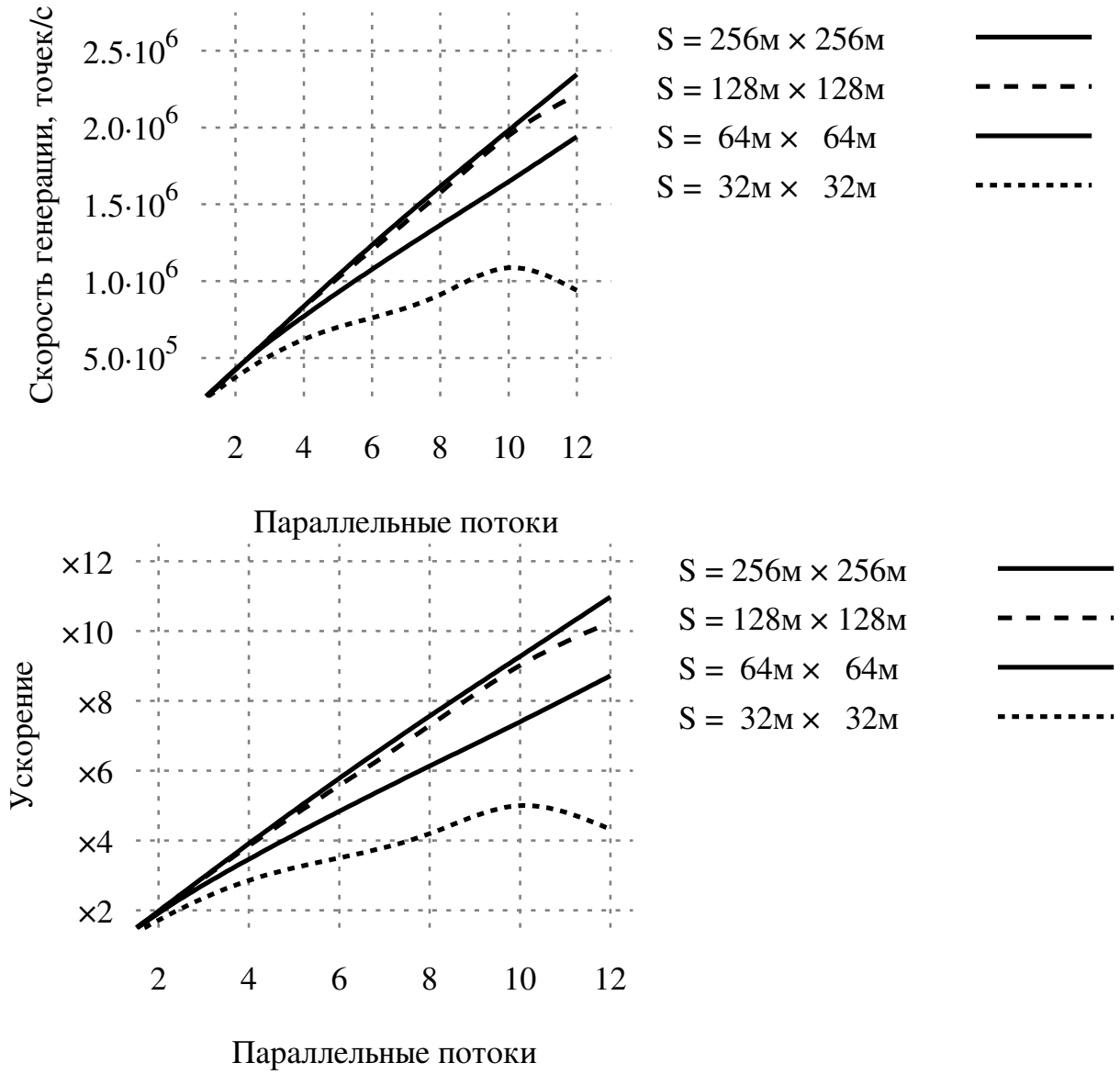

Рис. 6: Скорость генерации взволнованной поверхности на многопроцессорной системе для типовых размеров реализации (сверху). Масштабируемость (относительное ускорение при увеличении количества процессоров) программной реализации на многопроцессорной системе для типовых размеров реализации (снизу). Временная протяженность 512 с.

(расчета коэффициентов авторегрессионной модели) была выбрана GotoBLAS и основанная на ней LAPACK, для непрерывной аппроксимации поля волновых чисел использовалась библиотека CGAL [25] и для статистической проверки интегральных характеристик реализации взволнованной поверхности использовалась библиотека GSL [26]. В случае GotoBLAS эффективность библиотеки показана в работах [27, 28], для других библиотек эффективность не является важной, и они были выбраны, исходя из удобства их использования.

Алгоритм распределения нагрузки. Наиболее разработанным и широко применяемым подходом к распределению нагрузки на вычислительную систему является разбиение данных на однородные части (или разбиение задачи на однородные подзадачи) с последующим распределением их между отдельными ядрами вычислительного устройства или узлами кластера, однако такой подход не всегда работает эффективно. Во-первых, часто общее количество частей, на которые разбиваются входные данные, диктуется не архитектурой и конфигурацией вычислительной системы, а самой задачей и присущими ей ограничениями, и такое распределение не всегда эффективно с точки зрения вычислительной машины: количество частей оказывается либо слишком большим по сравнению с количеством процессоров, работающих параллельно, что ведет к увеличению накладных расходов на обмен данными, либо слишком маленьким, что не позволяет эффективно использовать все доступные вычислительные ядра. Во-вторых, сам характер деления входных данных на части может стать причиной появления неоднородности в размерах различных частей и дисбаланса нагрузки на отдельные вычислительные ядра системы. В-третьих, поскольку в вычислительной системе процессор — не единственное устройство, справляющееся с нагрузкой, и существуют другие компоненты, участвующие в вычислениях (такие как векторные ускорители, видеокарты и устройства хранения), то окончательная производительность системы и время решения конкретной задачи зависят от производительности не только процессоров, но и всех устройств, принимающих участие в вычислениях. Таким образом, учет только лишь процессоров при распределении нагрузки на вычис-

лительную систему является лишь первым приближением к решению задачи о достижении высокой производительности, и учет всех компонент системы позволит улучшить этот результат.

Для учета производительности всех компонент вычислительной системы и неоднородности различных частей, на которые делятся входные и выходные данные, распределение нагрузки можно провести в два этапа. На первом этапе часть входных данных (или подзадача) направляется на соответствующее устройство, на которое предполагается произвести нагрузку, например, видеокарту или процессор; если же предполагается произвести нагрузку на устройство хранения, то подзадача направляется на процессор или процессорное ядро, специально выделенное под такой тип нагрузки. На втором этапе, когда тип устройства уже выбран, подзадача распределяется на одно из доступных в системе устройств данного типа. Несмотря на то, что на этом этапе подсистема в большинстве случаев является однородной (состоящей из устройств одного типа), для учета неоднородных подзадач необходим алгоритм распределения, который бы учитывал размер частей, на которые делится задача.

Такой алгоритм можно построить на основе алгоритма «заполнения» (англ. backfill), который широко применяется для распределения нагрузки на узлы вычислительного кластера, но для его эффективной работы в случае многопроцессорной системы нужно произвести определенные модификации. Для расчета времени решения задачи на кластере не существует надежного метода, и часто это время задается вручную перед отправкой задачи в очередь [29], что неприемлемо в случае многопроцессорной системы, и поэтому время решения отдельных подзадач необходимо предсказать. Для получения надежного предсказания можно воспользоваться любым подходящим статистическим методом и использовать время выполнения предыдущих подзадач в качестве исходных данных. Для учета неоднородной производительности устройств можно воспользоваться тем же методом, только в качестве исходных данных взять производительность устройства на предыдующих задачах (количество задач, завершенных в единицу времени).

Чтобы сократить накладные расходы, метод должен быть достаточно простым, и поэтому в тестах был использовано осреднение N последних значений характеристики. Использование статистических методов в случае многопроцессорной системы оправдано, поскольку в отличие от задач, решаемых на кластерах, время решения подзадач достаточно мало, чтобы статистические методы работали эффективно. В случае же кластерных систем ввиду очень большого времени решения одной задачи использование статистических методов не может дать надежный результат, и алгоритм «заполнения» работает эффективно для небольших задач [29].

Таким образом, распределение нагрузки осуществляется в два этапа: на первом этапе задача направляется на соответствующее ее типу нагрузки устройство, а на втором этапе она направляется на одно из выбранных устройств по алгоритму распределения. Сам же алгоритм является алгоритмом «заполнения» с модификациями, позволяющими автоматически предсказывать время выполнения задачи и производительность устройств.

Результаты тестирования. Программа, реализующая генерацию взволнованнной поверхности сбалансированна с точки зрения нагрузки на процессорные ядра, однако, как показали тесты, характеризуется высокой нагрузкой на устройства хранения. До проведения тестирования программа была реализована на OpenMP и для сравнения переписана в соответствии с разработанным подходом к распределению нагрузки, реализованным в виде отдельной библиотеки. Конфигурация оборудования, использованная в тестах, приведена в табл. 5. В результате две реализации были сопоставлены с точки зрения производительности.

В процессе экспериментов была измерена эффективность описанного метода распределения нагрузки, и он показал более высокую производительность в задаче генерации взволнованной поверхности (в задаче генерации большого объема данных) по сравнению с реализацией OpenMP. В результате предыдущих исследований было установлено, что реализация OpenMP имеет наилучшую производительность по сравнению с другими технологиями параллельного программиро-

Компонента	Подробности
Язык программирования	C++11
Библиотека потоков	C++11 STL threads
Библиотека атомарных операций	C++11 STL atomic
Подпрограммы замера времени	`clock_gettime(CLOCK_MONOTONIC)` `/usr/bin/time -f %e`
Компилятор	GCC 4.8.2
Опции компиляции	`-std=c++11 -O2 -march=native`
Операционная система	Debian 3.2.51-1 x86_64
Файловая система	ext4
Процессор	Intel Core 2 Quad Q9650
Частота процессора (ГГц)	3.00
Количество ядер	4
Емкость ОЗУ (ГБ)	8
Диск	Seagate ST3250318AS
Скорость диска (об./мин.)	7200

Таблица 5: Конфигурация многопроцессорной вычислительной системы.

вания [24], поэтому эксперимент заключался в сравнении ее производительности с производительностью нового метода на ряде входных данных. При каждом запуске варьировался только размер взволнованной поверхности. В результате эксперимента было установлено, что с увеличением размера поверхности увеличивается разрыв в производительности этих двух реализаций (см. рис. 7), а высокая производительность предложенного метода обуславливается наложением по времени фазы генерации взволнованной поверхности и фазы записи ее на диск (см. рис. 8). В реализации OpenMP такого наложения не происходит, и запись на диск начинается сразу после окончания генерации поверхности в отличие от новой реализации, в которой генерация и запись на диск заканчиваются почти одновременно. Таким образом, в программах, работающих с большим объемом данных, конвейеризация параллельных вычислительных фаз более эффективна, чем их последовательное выполнение и позволяет сбалансировать нагрузку не только на процессорные ядра, но и на дисковую подсистему.

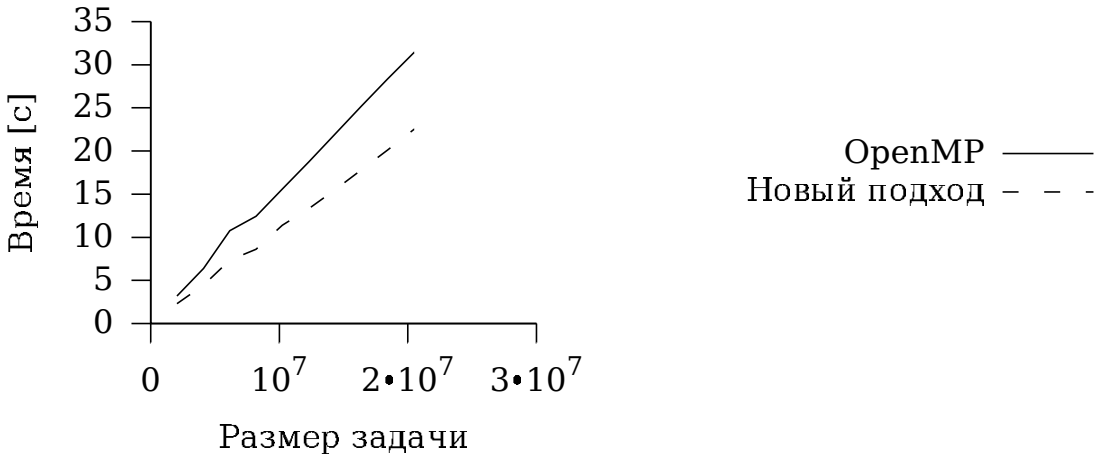

Рис. 7: Сравнение производительности реализаций программы на OpenMP и на разработанной технологии

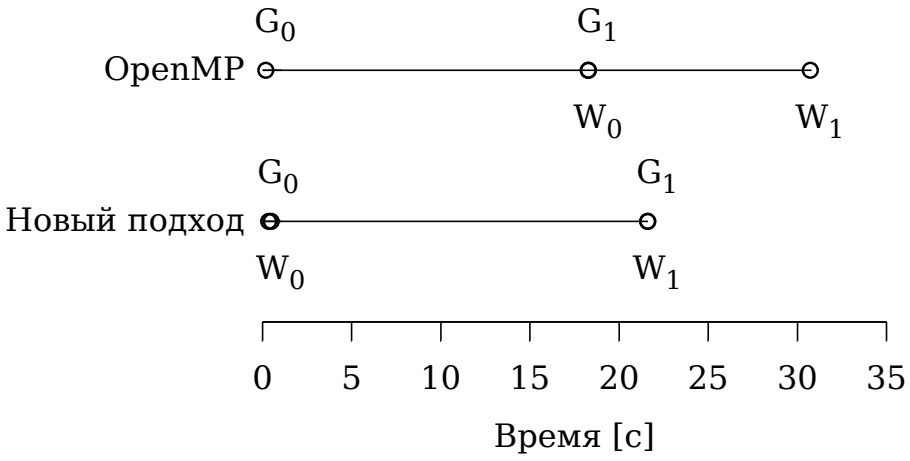

Рис. 8: Наложение параллельных вычислений на $[G_0, G_1]$ и записи данных на диск на $[W_0, W_1]$. В реализации OpenMP наложение отсутствует

Несмотря на то, что технология OpenMP содержит примитивы для создания конвейеров, соединить конвейером две распараллеленные фазы программы можно только вручную. Такое соединение можно реализовать с помощью синхронизированной очереди, в которую направляются сгенерированные части взволнованной поверхности, готовые к записи в файл. Используя директиву *omp section*, можно описать работу каждого из звеньев получившегося конвейера, однако реализовать параллельную обработку каждого из звеньев (или хотя бы первого) не представляется возможным, так как требуется поддержка вложенных директив *omp parallel*, что редко встречается в реализациях OpenMP. Таким образом, реализация описанного метода распределения нагрузки в рамках стандарта OpenMP затруднена.

Предложенный метод распределения нагрузки на многопроцессорную систему позволяет получить прирост производительности для приложений, считывающих и записывающих большой объем данных на диск, позволяет сбалансировать нагрузку на процессорные ядра вычислительной системы и назначить различные типы рабочей нагрузки разным процессорным ядрам, а также различным устройствам, в том числе дискам. В дальнейших исследованиях предполагается обобщить этот метод на распределенную вычислительную среду.

3.2. Комплекс программ для систем с распределенной памятью

Многие распределенные системы построены по принципу *субординации*: в каждом кластере выбирается главный узел, который управляет очередью задач решаемых на кластере и мониторингом их выполнения. Роль главного узла может задаваться как *статически*, путем выделения конкретного физического узла под нее, так и *динамически*, путем избрания какого-либо из узлов кластера галвным. В первом случае отказоустойчивость обеспечивается посредством резервирования дополнительного свободного узла, который займет место главного в случае отказа оборудования. Во втором случае отказоустойчивость обеспечивается выбором нового главного узла из оставшихся в случае отказа текущего. Несмотря на то что динамический распределение ролей требует наличия распределенного алгоритма, этот подход становится все более и более популярным, поскольку не требует наличия простаивающих резервных узлов на случай отказа главного узла.

Алгоритмы выбора лидера (которые иногда называют алгоритмами распределенного консенсуса) являются частными случаями волновых алгоритмов. В [30] Тель определяет их как алгоритмы, в которых событие завершения программы предваряется хотя бы одним каким-либо другим событием, происходящем в *каждом* параллельном процессе. Волновые алгоритмы не определены для анонимных сетей, т.е. они работают только с теми параллельными процессами, которые могут себя уникально идентифицировать. Однако, количество процессов, которых затрагивает «волна» может быть определено по мере выполнения алгоритма. В рамках распределенных систем это означает, что волновые алгоритмы подходят для вычислительных кластеров с динамически меняющимся количеством узлов, так что включение и выключение отдельных узлов не влияет на работу алгоритма.

Подход к динамическому выбору главного узла, исследованнный в данной работе, не использует волновые алгоритмы, а значит не требует опроса квору-

ма узлов для выбора лидера. Вместо этого каждый узел кластера составляет список других узлов подсети, в которой он находится, и простым алгоритмом преобразует список в *древовидную иерархию* с заданным значением ветвления (максимальным количеством подчиненных вершин). Затем узел определяет уровень иерархии, на котором он находится и пытается соединиться с вышестоящими узлами, чтобы стать их подчиненным. Сначала он проверяет близко расположенные к нему узлы, а потом все остальные узлы вплоть до вершины иерархии. Если вышестоящих узлов нет или с ними невозможно соединиться, то этот узел становится главным.

Древовидная иерархия узлов подсети определяет отношение строгого порядка на множестве всех узлов кластера. С технической точки зрения любая функция может быть выбрана для присвоения узлу подсети номера в списке, однако, на практике эта функция должна быть достаточно гладкой вдоль временной оси и иметь лишь редкие скачки: быстрые изменения в структуре иерархии узлов могут привести постоянной передачи роли главного узла от одного узла к другому, что сделает кластер неуправляемым. Простейшей такой функцией является позиция IP-адреса узла в диапазоне всех IP-адресов подсети.

Основной особенностью алгоритма является многоуровневая субординация, т.е. выбор сразу нескольких лидеров в рамках одной подсети в зависимости от значения ветвления иерархии. Это позволяет делать локальные изменения в структуре иерархии, не затрагивающие всех узлов кластера, и определить адрес потенциального главного узла еще до начала алгоритма. Подход отличается от волновых алгоритмов наличием сразу нескольких лидеров в одной подсети, использованием IP-адресов узлов в качестве уникального идентификатора и критерия ранжирования, а также отсутствием какой-либо предварительной коммуникации между узлами, которая нужна для определения их ранга.

Алгоритм *субординации*, исследованнный в данной работе, позволяет объединить узлы вычислительного кластера в распределенную систему без какой-либо предварительной конфигурации, только лишь установив и запустив соответ-

свующие сервисы на каждом из узлов. При выходе из строя одного из узлов эти сервисы способны самостоятельно найти исправный узел и стать его подчиненным, или же занять вершину иерархии. Такая автономность работы выгодно отличает субординацию от традиционных подходов к управлению вычислительным кластером, который включает в себя ручную настройку главного и подчиненных узлов, а также резервного узла для обеспечения отказоустойчивости.

В отличие от других алгоритмов выбора лидера [31–33], алгоритм субординации не предназначен для управления одновременным обновлением записей в распределенной базе данных несколькими параллельными процессами; вместо этого основной областью применения алгоритма служит распределение нагрузки на большое количество узлов кластера. Обычно, кластер управляет одним главным узлов (или 1-2 узлами для обеспечения отказоустойчивости), который собирает данные мониторинга, ведет учет потребленных пользователями ресурсов, позволяет изменять настройки всего кластера и ставит задачи в очередь для запуска. Для больших кластеров одного главного узла может быть недостаточно, чтобы справиться с нагрузкой, и в этом случае введение подчиненных узлов, управляющих отдельными частями кластера, может решить эту проблему.

Алгоритм субординации предполагает, что IP-адрес узла изменяется редко (при этом один и тот же IP-адрес необязательно должен быть закреплен за определенным узлом). Практика показывает, что это предположение выполняется для кластеров, однако это может стать проблемой для виртуальных кластеров, создаваемых на ресурсах облачных провайдеров: в таких кластерах IP-адрес может меняться произвольно с сохранением DNS-имени узла. Использование алгоритм субординации в такой среде может привести к постоянному переназначению ролей узлов кластера, что не позволит эффективно распределять нагрузку.

Суммируя вышесказанное, алгоритм субординации не подходит для сред, в которых IP-адреса меняются часто, основной сферой применения алгоритма является распределение нагрузки на вычислительный кластер и алгоритм не требует какой-либо предварительной конфигурации.

Построение древовидной иерархии. Субординация на множестве $\mathcal{N}$ узлов одной подсети определяется как

$$\forall n_1 \forall n_2 \in \mathcal{N}, \forall f \colon \mathcal{N} \to \mathcal{R}^n \Rightarrow (f(n_1) < f(n_2) \Leftrightarrow \neg(f(n_1) \geq f(n_2))),$$

где f — отображение узла на его идентификационный номер, и $<$ — оператор, определяющий отношение строго порядка на множестве $\mathcal{R}^n$. Функция f присваивает узлу его порядковый номер, а оператор $<$ делает этот номер уникальным.

Простейшее отображение f ставит в соответствие каждому узлу подсети позицию его IP-адреса в диапазоне всех адресов подсети. Без преобразования к древовидной структуре (когда в подсети выбирается только один лидер) рабочий узел, адрес которого занимает наименьшую позицию в диапазоне, становится главным. Если адрес узла занимает первую позицию в диапазоне, то для него невозможно выбрать лидера, и он будет находится на вершине иерархии вплоть до выхода из строя. Несмотря на то что идентификацию узлов на основе их IP-адресов легко реализовать в программе, такой подход устанавливает искусственную зависимость роли главного узла от IP-адреса. Тем не менее, этот подход полезен для первичного объединения узлов в кластер, когда более сложные методы идентификации узлов неприменимы.

Для того чтобы алгоритм субординации масштабировался на большое количество узлов, структуру диапазона адресов подсети необходимо сделать древовидной. В древовидной иерархии каждый узел идентифицируется уровнем l иерархии, на котором он находится и отступом o, который равен порядковому номеру узла на его уровне. Значения уровня и отступа определяются из следующей задачи оптимизации.

$$n = \sum_{i=0}^{l(n)} p^i + o(n), \quad l \to \min, \quad o \to \min, \quad l \geq 0, \quad o \geq 0$$

где n — позиция IP-адреса узла в диапазоне IP-адресов подсети и p — значение ветвления (масимальное количество подчиненных, которых может иметь узел). Лидер узла на уровне l с отступом o будет иметь уровень $l - 1$ и отступ $\lfloor o/p \rfloor$. Расстояние между любыми двумя узлами в иерархии, адреса которых занимают позиции i и j в диапазоне определяется как

$$\langle \text{lsub}(l(j), l(i)), \quad |o(j) - o(i)/p| \rangle,$$

$$\text{lsub}(l_1, l_2) = \begin{cases} \infty & \text{if } l_1 \geq l_2, \\ l_1 - l_2 & \text{if } l_1 < l_2. \end{cases}$$

Расстояние имеет составную запись, чтобы при определении близости двух узлов уровень иерархии учитывался в первую очередь.

Для выбора лидера каждый узел ранжирует все узлы подсети в соответствии с $\langle l(n), o(n) \rangle$ и, используя формулу для определения расстояния, выбирает ближайший к потенциальному лидеру узел, имеющий наименьший ранг. Это позволяет пропустить IP-адреса выключенных и просто несуществующих узлов, но для разреженных сетей, в которых некоторые IP-адреса из середины списка не закреплены за работающими узлами, сбалансированность дерева не гарантируется.

Поскольку узлу для выбора лидера нужно соединиться только с узлом, адрес которого известен заранее, то алгоритм субординации хорошо масштабируется на большое количество узлов. Соединение с другими узлами из ранжированного списка происходит только в том случае, если соединение с узлом из начала списка прерывается. Таким образом, если адреса рабочих узлов расположены плотно в диапазоне адресов подсети, каждый узел устанавливает соединение только со своим узлом-лидером, и ресурсоемкого и неэффективного сканирования всей сети каждым узлом не происходит.

Производительность алгоритма субординации. Платформа, на которой осуществлялось тестирование, состоит из одного многопроцессорного узла, на котором развертывается виртуальный кластер из заданного количества узлов с

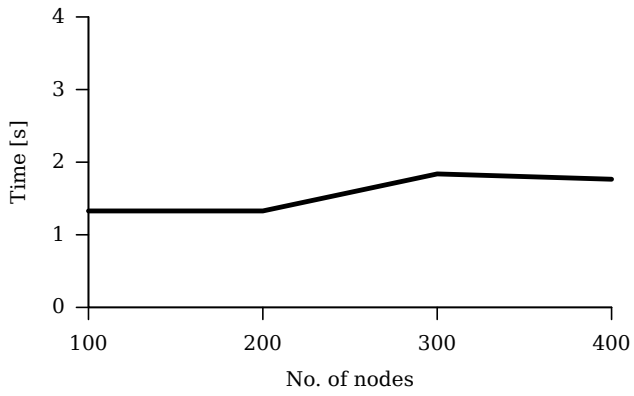

Рис. 9: Зависимость времени объединения узлов в кластер в зависимости от их количества.

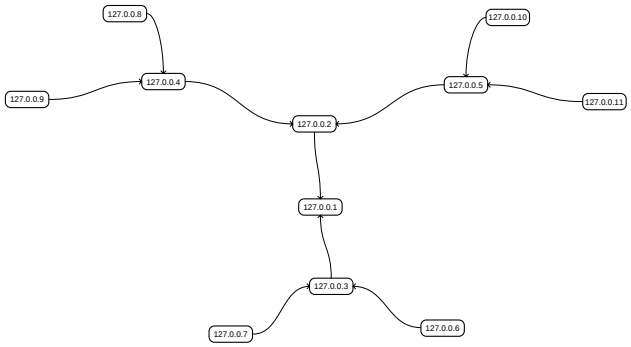

Рис. 10: Древовидная иерархия для 11 узлов.

помощью пространств имен Linux. Похожий подход используется в [34–36], где обосновывается целесообразность его применения для проведения экспериментов на виртуальных кластерах и сопоставляются результаты некоторых из них с реальными кластерами. Преимуществом данного подхода является низкие требования к оборудованию, на котором проводятся эксперименты, а также отсутствие влияния внешних процессов, выполняющихся параллельно с экспериментами.

Тестирование производительности заключалось в построении графика зависимости времени, затрачиваемого на объединение узлов в кластер, от количества узлов. В процессе эксперимента любое изменение иерархии записывалось в файл и по прошествии 30 сек. все процессы вынужденно останавливались системой. Пробные запуски показали, что одновременный запуск более 100 виртуальных узлов искажал результаты, поэтому для этого эксперимента были использованы дополнительные физические узлы, на каждом из которых создавалось по 100 виртуальных. Эксперимент показал, что объединение от 100 до 400 узлов в кластер занимает в среднем 1,5 секунды (см. рис. 9). Для полностью физического кластера это значение может увеличиться. Пример древовидной иерархии, полученной при запуске на 11 узлах представлен на рис. 10.

Обеспечение отказоустойчивости. Отказы узлов распределенной системы можно разделить на три типа: отказ подчиненного узла, отказ главного узла и отказ одновременно всех узлов (отключение электричества). Для того чтобы

запущенная на кластере задача могла продолжиться после отказа подчиненного узла, для нее периодически создаются и записываются в надежное хранилище контрольные точки восстановления. При создании контрольной точки все параллельные процессы задачи временно останавливаются, образ памяти, выделенной операционной системой для процессов задачи копируется на диск, и выполнение задачи продолжается в нормальном режиме. Для того чтобы отказ главного узла не повлиял на работу кластера, состояние сервисов, запущенных на нем, непрерывно копируется на резервный узел, который становится главным при отказе. При незапланированном отключении электричества состояние всех запущенных на момент отказа задач восстанавливается из контрольных точек восстановления.

Оптимизации работы контрольных точек восстановления посвящено большое количество работ [37], а альтернативным подходам уделяется меньше внимания. Обычно высокопроизводительные приложения используют передачу сообщений для обмена данными между параллельными процессами и хранят свое текущее состояние в глобальной памяти, поэтому не существует способа перезапустить завершившийся процесс, не записав образ всей выделенной для него памяти на диск. Обычно общее число процессов фиксированно и задается планировщиком, и в случае отказа перезапускаются сразу все процессы. Существуют некоторые обходные решения, которые позволяют перезапустить только часть процессов [38], восстановив их на других узлах, однако это может привести к перегрузке, если на этих узлах уже запущены другие задачи. Теоретически, перезапуск процесса необязателен если задача может быть продолжена на меньшем количестве узлов, но библиотека передачи сообщений не позволяет изменять количество параллельных процессов во время работы программы, и большинство программ все равно предполагают, что это значение является константой, и используют его для распределения нагрузки между узлами. Таким образом, не существует простого способа обеспечения отказоустойчивости на уровне библиотеки передачи сообщений кроме как путем перезапуска всех параллельных процессов из контрольной точки восстановления.

В то же время, существует возможность продолжить выполнение задачи на меньшем количестве узлов, чем было изначально выделено под нее планировщиком. В этом случае нагрузка должна быть динамически перераспределена между оставшимися узлами. Несмотря на то что динамическое распределение нагрузки было реализовано в поверх библиотеки передачи сообщений в ряде работ [39, 40], оно никогда не применялось в задаче обеспечения отказоустойчивости. В этом разделе исследуются методы обеспечения отказоустойчивости при выходе из строя подчиненных и главных узлов и показывается, как приемы объектно-ориентированного программирования могут быть использвоаны для сохранения минимального состояния программы, необходимого для ее перезапуска, в иерархии объектов, а не в глобальных и локальных переменных.

Иерархия объектов вычисления. Для распределения нагрузки используется узлы кластера объединяются в древовидную иерархию. Нагрузка распределяется между непосредственными соседями узла, так что при запуске задачи на подчиненном узле главный узел также получают часть нагрузки. Это делает систему симметричной и легкой в обслуживании: на каждом узле установлен один и тот же набор программного обеспечения, что позволяет заменить один узел другим при выходе из строя первого. Похожее архитектурное решение используется в хранилищах типа «ключ-значение» [41, 42] для обеспечения отказоустойчивости при выходе из строя одного из узлов, однако автору неизвестны планировщики задач, которые используют данный подход.

Каждая программа, запущенная поверх ирерахии узлов состоит из *вычислительных объектов* — объектов, которых содержат данные и код для их обработки. Для эффективного использования параллелизма, предоставляемого кластером и многопроцессорной машиной, объект может создать подчиненные (дочерние) объекты, которые система автоматически распределит сначала между доступными процессорными ядрами, затем между подчиненными узлами кластера. Сама программа также является вычислительным объектом, который либо решает при-

кладную задачу последовательно, либо создает подчиненные объекты для параллельного решения.

В отличие от функции `main` в программах на основе библиотеки передачи сообщений, первый вычислительный объект выполняется только на одном узле, а дополнительные узлы используются либо при переполнении очереди объектов текущего узла, либо при явном указании в коде программы. Такая архитектура позволяет использовать произвольное количество узлов для запуска задачи и динамически менять это количество во время ее выполнения. Похожий принцип выполнения задач используется в системах обработки больших объемов данных [43, 44] — при запуске задаче не требуется указывать количество узлов, вместо этого система сама выбирает узлы, на которых будет выполняться задача, в зависимости физического расположения входных файлов.

С математической точки зрения вычислительный объект K может быть определен как векторнозначный функционал, отображающий один вычислительный объект на n-компонентный вектор вычислительных объектов:

$$K(f) : \mathbb{K} \to \mathbb{K}^n \qquad \mathbb{K}^n = \{f : \mathbb{K} \to \mathbb{K}^n\}.$$

Спецальный объект $\mathbb{O} : \mathbb{K} \to \mathbb{K}^0$ используется для остановки рекурсии, и передается в качестве аргумента главному (первому созданному) объекту программы. Аргумент функционала интерпретируется следующим образом.

1. Если текущий объект является только что созданным объектом, то аргумент функционала — это главенствующий над ним объект (родитель).

2. Если текущий объект является родителем объекта, который его породил или родителем какого-либо другого объекта, то аргумент функционала — объект, которые его породил.

Объекты обрабатываются в цикле, который начинается с вызова функции главного объекта программы, затем вызываются функции всех порожденных им

объектов. Цикл продолжается до тех пор пока функция какого-либо объекта не вернет $\mathbb{O}$. Поскольку вызов функции может породить сразу несколько объектов, они выполняются параллельно, что приводит к быстрому заполнению пула объектов, которые можно выполнять в произвольном порядке. Несколько потоков одновременно выбирают из пула объекты для обработки, и при переполнении пула объеты могут быть переданы на другие узлы кластера без явного указания в исходном коде программы.

Вычислительные объекты реализованы в виде замыканий (функторов) — объектов-функций, которые сохраняют в себе аргументы, ссылку на породивший их объект и данные из предметной области задачи. Данные обрабатываются либо при выполнении объекта, либо для параллельной обработки создаются дочерние объекты. Когда обработка завершена, родительский объект вызывается с дочерним объектов в качестве аргумента для сбора результатов обработки.

Выход из строя одного узла. Наиболее распространенная стратегия при выходе из строя подчиенного узла — перезапуск выполнявшихся на нем объетов на рабочих узлах. Этой стратегии следует язык Erlang для перезапуска подчиненных процессов [45]. Для того что реализовать этот метод в рамках иерархии вычислительных объектов необходимо сохранять каждый объект передаваемый на другие узлы кластера. В случае отказа одного из узлов, на которые были переданы объекты, соответствующие их копии извлекаются из очереди на перераспределяются между оставшимися узлами без какой-либо дополнительной обработки. Если больше узлов не осталось, то объекты перенаправляются в локальную очередь. В отличие от «тяжеловесного» метода контрольных точек восстановления, древовдиная иерархия узлов в паре с иерархией объектов позволяет автоматически продолжить выполнение программы при выходе из строя одного из узлов без перезапуска каких-либо процессов задачи.

Возможная стратегия при выходе из строя узла, на котором хранится главный вычислительный объект задачи, заключается в копировании этого объекта на резервный узел и синхронизировать любые изменения между двумя копиями

объекта посредством распределенных транзакций. Однако, эта стратегия не соотносится с асинхронностью вычислительных ядер и слишком сложна в реализации. На практике, оказывается, что главный объект программы обычно не создает больше одного дочернего объекта, каждый из которых представляет собой последовательный шаг вычислений (внутри которого может быть, а может не быть параллельных этапов). Поскольку шаги последовательны, то одновременно может существовать не более одного дочернего объекта, что позволяет упростить синхронизацию состояния главного объекта программы. Для этого главный объект передается на подчиненный узел вместе со своим дочерним объектом. Тогда при выходе из строя узла, на котором была запущена программа, резервный узел автоматически восстанавливает состояние главного объекта из копии, когда дочерний объект завершает свою работу.

Описанный выше подход предназначен только для объектов, у которых нет объекта-родитея и которые создают по одному дочернему объекту за раз. Это означает, что метод работает как контрольная точка восстановления, которая сохраняет состояние только между последовательными шагами вычислений (когда оно занимает минимальный объем памяти) и которая для сохранения состояния использует оперативную память другого узла кластера, а не диск.

Программная реализация. Из соображений эффективности методы обеспечения отказоустойчивости были реализованы во фреймворке на языке C++: с точки зрения автора язык C слишком низкоуровневый для написания распределенных программ, а использование языка Java влечет за собой накладные расходы, и непоулярно в высокопроизводительных вычислениях. Для того чтобы использовать фреймворка без планировщика задач необходимо создать сервис, который бы автоматически обновлял состояние древовидной иерархии узлов и предоставлял программный интерфейс для запуска задач на ней. На данный момент фреймворк запускает сервис и приложение в одном процессе. Фреймворк называется «Фабрика» и находится на этапе проверки концепции.

Таблица 6: Конфигурация кластера.

CPU	Intel Xeon E5440, 2.83GHz
RAM	4Gb
HDD	ST3250310NS, 7200rpm
Кол-во узлов	12
Кол-во ядер на узел	8

Результаты тестирования Методы отказоустойчивости были протестированы на физическом кластере (см. 6) на примере программы, генерирующей взволнованную морскую поверхность, подробно описанной в [46–49] и в данной работе. Программа состоит из серии фильтров, каждый из которых применяется к результату работы предыдущего. Некоторые из фильтров применяются параллельно, так что вся программа состоит из последовательно выполняющихся больших шагов, некоторые из которых внутри реализованы параллельно из соображений эффективности. Только наиболее ресурсоемкий этап программы (генерация взволнованной морской поверхности) выполняется параллельно на всех узлах, другие этапы выполняются параллельно на всех процессорных ядрах главного узла.

Программа была переписана под новую версию фреймворка, что потребовало лишь небольших изменений исходного кода для корректной обработки выхода из строя узла с главным объектом: главный объект был помечен, чтобы фреймворк смог передать его на подчиненный узел. Другие изменения исходного кода были связаны с изменением программного интерфейса фреймворка. Таким образом, обеспечение отказоустойчивости, в основном, прозрачно для программиста и требует лишь помечивания главного объекта для его репликации на резервный узел.

В ряде экспериментов провзводительность новой версии программы была измерена при выходе из строя различных типов узлов во время выполнения программы (номера пунктов соответствуют номерам графиков рис. 11):

1. без выхода из строя узлов,

2. выход из строя подчиненного узла (на котором генерируется часть взволнованной поверхности),
3. выход из строя главного узла (на котором запускается программа),
4. выход из строя резервного узла (на который копируется главный объект программы).

Древовидная иерархия узлов со значением ветвления равного 64 использовалась в экспериментах, для того чтобы удостовериться, что все подчиненные узлы соединены с первым узлом подсети кластера. При каждом запуске программы главный объект запускался не на главном узле, чтобы оптимально отобразить иерархию объектов на иерархию узлов (наложить одну на другую). Узел-жертва выводился из строя по прошествии фиксированного временного интервала после запуска программы равного примерно $1/3$ времени работы программы на одном узле. Способ запуска для каждого эксперимента представлен в табл. 7 («корень» и «лист» относятся к положению узла в древовидной иерархии). Результаты экспериментов приведены на рис. 11 и рис. 12.

Графики 2 и 3 на рис. 11 показывают, что производительность в случае выхода из строя главного и подчиненного узлов примерно одинакова. В случае отказа главного узла резервный узел сохраняет копию главного объекта и восстанавливает главный объект из нее, когда не обнаруживает, что главный узел вышел из строя. В случае отказа подчиненного узла, главный узел перераспределяет невернувшиеся объекты между оставшимися подчиненными узлами. В обоих случая состояние главного объекта программы не теряется, а значит не тратится время на его восстановление, что объясняет схожую производительность.

График 4 на рис. 11 показывает, что производительность в случае выхода из строя резервного узла гораздо ниже, чем в других случаях. Это происходит, потому что главный узел сохраняет состояние только текущего последовательного этапа программы, в то время как резервный узел не только хранит копию этого состояния, но и выполняет параллельные части этапа вместе с другими подчи-

Таблица 7: Параметры экспериментов.

Номер эксп.	Главный узел	Узел-жертва	Время до выхода из строя, сек.
1	корень		
2	корень	лист	10
3	корень	лист	10
4	корень	лист	10

ненными узлами. Так что, когда резервный узел выходит из строя, главный узел начинает выполнение текущего этапа с самого начала.

Для оценки количества времени, которое теряется при выходе из строя одного из узлов, можно поделить общее время работы программы со сбоем на время работы программы без сбоев, но с количеством узлов минус один. Это отношение представлено на рис. 12. Разница в производительности в случае выхода из строя главного узла и подчиненного узла находится в пределах 5%, а в случае выхода из строя резервного узла — в пределах 50% для количества узлов меньше 6[2]. Разница в 50% больше, чем $1/3$ времени работы программы, после которого происходит сбой, однако отказ резервного узла требует некоторого времени, чтобы быть обнаруженным другими узлами. Сбой узла обнаруживается только тогда, когда подчиненный объект завершает свое выполнение и пытается вернуться на исходный узел к родителю. Мгновенное обнаружение сбоя узла требует остановки выполнения объектов, что может быть неприменимо для программ со сложной логикой.

Результаты экспериментов позволяют сделать вывод о том, что *не важно, вышел ли из строя главный узел или подчиненный, итоговое время работы программы будет примерно равно времени ее работы без сбоев, но с уменьшенным на единицу количеством узлов*, однако, в случае выхода из строя резервного узла теряется гораздо больше времени.

[2]Измерение разницы для большего количества узлов не имеет смысла, поскольку программа завершается еще до наступления сбоя.

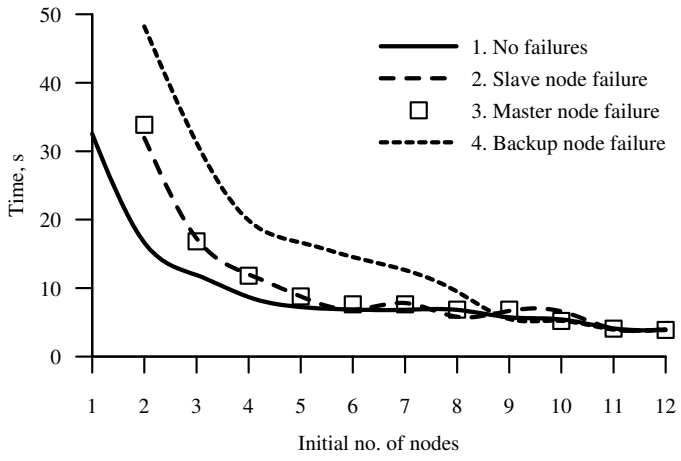

Рис. 11: Производительность программы генерации взволнованной морской поверхности при различных типах сбоев узлов.

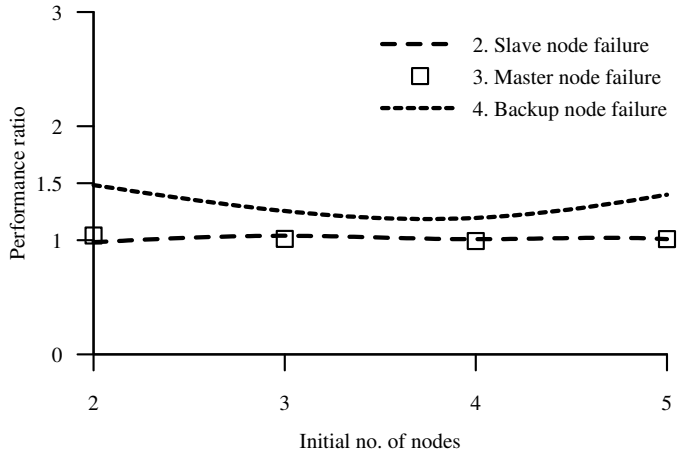

Рис. 12: Замедление программы генерации взволнованной морской поверхности при различных типах сбоев по сравнению с запуском без сбоев но с уменьшенным на единицу количеством узлов.

Выводы по результатам тестирования. Проведенные эксперименты показывают, что параллельной программе необходимо иметь несколько последовательных этапов выполнения, чтобы сделать ее устойчивой к сбоям узлов. Несмотря на то что вероятность сбоя резервного узла меньше вероятности сбоя одного из подчиненных узлов, это не повод потерять все данные, когда выполнявшаяся несколько дней задача почти завершилась. В общем случае, чем больше последовательных этапов вычислений содержит программа, тем меньше времени потеряется в случае сбоя резервного узла, и, аналогично, чем больше параллельных частей содержит каждый этап, тем меньше времени потеряется при сбое главного или подчиненного узла. Другими словами, *чем больше количество узлов, на которое масштабируется программа, тем она более устойчива к сбою оборудования*.

В проведенных экспериментах узел, на котором запускается программа, выбирается вручную, чтобы совместить иерархию объектов и иерархию узлов, однако, такой подход не применим в реальных системах. Разработанный фреймворк должен делать это автоматически и эффективно распределять нагрузку между подчиненными узлами в незваисимости от местоположения главного узла в иерархии: выделение одного и того же узла в качестве главного для запуска нескольких приложений уменьшает общую отказоустойчивость системы.

Хотя это неочевидно из экспериментов, Фабрика не только обеспечивает отказоустойчивость, но и позволяет автоматически вводить новые узлы в кластер и распределять на них часть нагрузки уже запущенных программ. В контексте фреймворка этот процесс тривиален, поскольку не требует перезапуска незавершившихся объектов и копирования их состояния, и не изучался экспериментально в данной работе.

Теоретически отказоустойчивость, основанная на иерархии может быть реализована поверх библиотеки передачи сообщений без потери общности. Хотя использование незагруженных узлов заместо вышедших из строя в рамках такой библиотеки представляет определенную сложность, поскольку количество узлов, на которых запущена программа фиксировано, но выделение достаточно боль-

шого количества узлов для программы будет достаточно для обеспечения отказоустойчивости.

Слабым местом описанных методов является период времени, начиная с отказа главного узла и заканчивая обнаружением сбоя подчиненным узлом. Если до момента восстановления главного объекта из копии резервный узел выходит из строя, то состояние выполения программы полностью теряется без возможности его восстановить, кроме как перезупаском с самого начала. Длина этого опасного промежутка времени может быть уменьшена, но исключить его полностью невозможно. Этот результат согласуется с исследованиями теории «невыполнимости», в рамках которой доказывается невозможность распределенного консенсуса с хотя бы одним процессом, дающим сбой [50] и невозможность надежной передачи данных в случае сбоя одного из узлов [51].

Заключение

Итоги исследования. В изучении возможностей математического аппарата для имитационного моделирования морского волнения, выходящего за рамки линейной теории волн, были достигнуты следующие основные результаты.

- Разработана модель ветрового волнения для генерации волн произвольных амплитуд.
- Разработан гибридный метод расчета давлений в двухмерной постановке, использующий как аналитические выражения, так и численное интегрирование по алгоритму быстрого преобразования Фурье.
- Разработанный метод позволил упростить программную реализацию комплекса, и свести реализацию алгоритма к параллельному вычислению большого количества преобразований Фурье.
- Предварительная апробация авторегрессионной модели и метода расчета давлений была произведена в пакете Large Amplitude Motion Program 4.

Перспективы дальнейших исследований. Хотя задача генерации взволнованной морской поверхности была решена в трехмерной постановке, для задачи расчета гидродинамических давлений было найдено аналитическое решение только в двухмерном случае. В трехмерной постановке для задачи было найдено аналитическое решение, но полученные формулы не были всесторонне исследованы. Таким образом, дальнейшие исследования предполагают всестороннее исследование аналитического решения для трехмерный случая, а также создание виртуального полигона на основе авторегрессионной модели и метода расчета давлений по результатам апробации в пакете Large Amplitude Motion Program 4.

Выводы

Результаты исследования позволяют сделать вывод о том, что задача вычисления давлений под реальной морской поверхностью может быть решена аналитически в двухмерной постановке, минуя предположения линейной теории волн и теории волн малой амплитуды. Это аналитическое решение в паре с авторегрессионной моделью ветрового волнения, способной генерировать волны произвольных амплитуд, может быть использовано для расчета влияния колебаний волн на поведение динамического объекта в открытом море, и дает более точные результаты чем аналогичное решение для волн малых амплитуд. Результаты проведенных экспериментов позволяют сделать вывод о том, что как генерация взволнованной поверхности так и расчет гидродинамических давлений могут быть реализованы эффективно с использование алгоритмов быстрого преобразования Фурье, а распределенная система научных расчетов может быть использвана для длительных сессий имитационного моделирования. Разработанный в работе математический аппарат и его программная реализация могут стать основой виртуального полигона, предназанченного для расчетов динамики морских объектов.

Список сокращений и условных обозначений

АР	авторегрессионная модель
АКФ	автоковариационная функция
НБП	нелинейное безынерционное преобразование
ГУ	граничное условие
БПФ	быстрое преобразование Фурье
ГПСЧ	генератор псевдослучайных чисел
ДУЧП	дифференциальное уравнение в частных производных

Список иллюстраций

1. Вид плотности распределения (14) волновых аппликат при различных значениях асимметрии γ_1 и эксцесса γ_2. 28
2. Вид плотности распределения (15) волновых аппликат при различных значениях коэффициента асимметрии α. 28
3. Поле потенциала скорости прогрессивной волны $\zeta(x, y, t) = \cos(2\pi x - t/2)$. Поле, полученное по формуле (10) (слева) и по формуле линейной теории волн (справа). 33
4. Сравнение полей скоростей на поверхности моря, полученных по общей формуле (u_1) и формуле для волн малой амплитуды (u_2). Поле скоростей для поверхности волн малой амплитуды (слева) и большой амплитуды (справа). 34
5. Схема конвейера обработки данных, реализующего генерацию взволнованной морской поверхности по АР модели. 36
6. Скорость генерации взволнованной поверхности на многопроцессорной системе для типовых размеров реализации (сверху). Масштабируемость (относительное ускорение при увеличении количества процессоров) программной реализации на многопроцессорной системе для типовых размеров реализации (снизу). Временная протяженность 512 с. 39
7. Сравнение производительности реализаций программы на OpenMP и на разработанной технологии 44
8. Наложение параллельных вычислений на $[G_0, G_1]$ и записи данных на диск на $[W_0, W_1]$. В реализации OpenMP наложение отсутствует 44
9. Зависимость времени объединения узлов в кластер в зависимости от их количества. 51
10. Древовидная иерархия для 11 узлов. 51

11. Производительность программы генерации взволнованной морской поверхности при различных типах сбоев узлов. 60
12. Замедление программы генерации взволнованной морской поверхности при различных типах сбоев по сравнению с запуском без сбоев но с уменьшенным на единицу количеством узлов. 60
13. Регрессия длин волн на их высоты 78
14. Скедастическая кривая длин и высот волн 78
15. Распределение высот волн 79
16. Распределение уклонов волн 79
17. Распределение длин волн 79
18. Распределение высот волн 79

Список таблиц

1. Значения параметров аппроксимации АКФ (2.1) для прогрессивных и стоячих волн ($A = 1, k = 1, \omega = 1, \sigma = 1$). 27
2. Формулы вычисления функций $\mathcal{D}_1(x, z)$ и $\mathcal{D}_2(x, z)$ из разд. 1.4, использующие нормировку для исключения неоднозначности определения дельта функции комплексного аргумента. 31
3. Конфигурация оборудования. 38
4. Время (с.) генерации взволнованной морской поверхности различными программными реализациями авторегрессионной модели. . . 38
5. Конфигурация многопроцессорной вычислительной системы. . . . 43
6. Конфигурация кластера. 57
7. Параметры экспериментов. 59

Список литературы

[1] Nonlinear time domain simulation technology for seakeeping and wave-load analysis for modern ship design. Authors' closure / Y S Shin, V L Belenky, W M Lin [и др.] // Transactions of Society of Naval Architects and Marine Engineers. 2003. Т. 111. С. 557–583.

[2] van Walree F, de Kat JO, Ractliffe AT. Forensic research into the loss of ships by means of a time domain simulation tool // International shipbuilding progress. 2007. Т. 54, № 4. С. 381–407.

[3] de Kat Jan O, Paulling J Randolph. Prediction of extreme motions and capsizing of ships and offshore marine vehicles // OMAE 2001-20th Conference on Offshore Mechanics and Arctic Engineering. 2001.

[4] Van Walree F. Development, validation and application of a time domain seakeeping method for high speed craft with a ride control system // Proceedings of the 24th Symposium on Naval Hydrodynamics. 2002.

[5] Schmitke RT, Whitten BT. SHIPMO: A FORTRAN PROGRAM TO PREDICT SHIP MOTIONS IN WAVES: Tech. Rep.: : 1981.

[6] Longuet-Higgins Michael S. The statistical analysis of a random, moving surface // Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences. 1957. Т. 249, № 966. С. 321–387.

[7] Degtyarev AB, Reed AM. Modelling of incident waves near the ship's hull (application of autoregressive approach in problems of simulation of rough seas) // Proceedings of the 12th International Ship Stability Work-shop. 2011.

[8] Бухановский А.В. Вероятностное моделирование полей ветрового волнения с учетом их неоднородности и нестационарности. Ph.D. thesis: СПбГУ. 1997.

[9] Kochin N., Kibel I., Roze N. Theoretical hydrodynamics [in Russian]. FizMatLit, 1966. C. 237.

[10] Degtyarev A., Gankevich I. Evaluation of hydrodynamic pressures for autoregression model of irregular waves // Proceedings of 11th International Conference on Stability of Ships and Ocean Vehicles, Athens. 2012. C. 841–852.

[11] Degtyarev Alexander B, Reed Arthur M. Synoptic and short-term modeling of ocean waves // International Shipbuilding Progress. 2013. T. 60, № 1-4. C. 523–553.

[12] Wolfram Research, Inc. Mathematica. Champaign, Illinois, 2016. (Version 10.4).

[13] Huang Norden E, Long Steven R. An experimental study of the surface elevation probability distribution and statistics of wind-generated waves // Journal of Fluid Mechanics. 1980. T. 101, № 01. C. 179–200.

[14] Рожков ВА. Теория вероятностей случайных событий, величин и функций с гидрометеорологическими примерами // С-Пб, Прогресс–Погода. 1996.

[15] Рожков Валентин Алексеевич, Трапезников Юрий Александрович. Вероятностные модели океанологических процессов. Гидрометеоиздат, 1990.

[16] Owen Donald B. Tables for computing bivariate normal probabilities // The Annals of Mathematical Statistics. 1956. T. 27, № 4. C. 1075–1090.

[17] Matsumoto Makoto, Nishimura Takuji. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator // ACM Transactions on Modeling and Computer Simulation (TOMACS). 1998. T. 8, № 1. C. 3–30.

[18] Matsumoto Makoto, Nishimura Takuji. Dynamic creation of pseudorandom number generators // Monte Carlo and Quasi-Monte Carlo Methods. 1998. T. 2000. C. 56–69.

[19] Discrete-time signal processing / Alan V Oppenheim, Ronald W Schafer, John R Buck [и др.]. Prentice hall Englewood Cliffs, NJ, 1989. Т. 2.

[20] Svoboda David. Efficient computation of convolution of huge images // Image Analysis and Processing–ICIAP 2011. Springer, 2011. С. 453–462.

[21] Pavel Karas, David Svoboda. Algorithms for efficient computation of convolution. INTECH Open Access Publisher, 2013.

[22] Pregel: a system for large-scale graph processing / Grzegorz Malewicz, Matthew H Austern, Aart JC Bik [и др.] // Proceedings of the 2010 ACM SIGMOD International Conference on Management of data / ACM. 2010. С. 135–146.

[23] Hama: An efficient matrix computation with the mapreduce framework / Sangwon Seo, Edward J Yoon, Jaehong Kim [и др.] // Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on / IEEE. 2010. С. 721–726.

[24] Degtyarev A., Gankevich I. Efficiency Comparison of Wave Surface Generation Using OpenCL, OpenMP and MPI // Proceedings of 8th International Conference “Computer Science & Information Technologies”. Yerevan, Armenia: 2011. P. 248–251.

[25] Fabri Andreas, Pion Sylvain. CGAL: The computational geometry algorithms library // Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems / ACM. 2009. С. 538–539.

[26] GSL GNU. Scientific Library. 2008.

[27] Goto Kazushige, Van De Geijn Robert. High-performance implementation of the level-3 BLAS // ACM Transactions on Mathematical Software (TOMS). 2008. Т. 35, № 1. С. 4.

[28] Goto Kazushige, Geijn Robert A. Anatomy of high-performance matrix multiplication // ACM Transactions on Mathematical Software (TOMS). 2008. Т. 34, № 3. С. 12.

[29] Zotkin Dinitry, Keleher Peter J. Job-length estimation and performance in backfilling schedulers // High Performance Distributed Computing, 1999. Proceedings. The Eighth International Symposium on / IEEE. 1999. С. 236–243.

[30] Tel Gerard. Introduction to distributed algorithms. Cambridge University press, 2000.

[31] Design and analysis of dynamic leader election protocols in broadcast networks / Jacob Brunekreef, Joost-Pieter Katoen, Ron Koymans [и др.] // Distributed Computing. 1996. Т. 9, № 4. С. 157–171.

[32] Stable leader election / Marcos K Aguilera, Carole Delporte-Gallet, Hugues Fauconnier [и др.] // Distributed Computing. Springer, 2001. С. 108–122.

[33] Romano Paolo, Quaglia Francesco. Design and evaluation of a parallel invocation protocol for transactional applications over the web // IEEE Transactions on Computers. 2014. Т. 63, № 2. С. 317–334.

[34] Lantz Bob, Heller Brandon, McKeown Nick. A network in a laptop: rapid prototyping for software-defined networks // Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks / ACM. 2010. С. 19.

[35] Reproducible network experiments using container-based emulation / Nikhil Handigol, Brandon Heller, Vimalkumar Jeyakumar [и др.] // Proceedings of the 8th international conference on Emerging networking experiments and technologies / ACM. 2012. С. 253–264.

[36] Heller Brandon. Reproducible Network Research with High-fidelity Emulation. Ph.D. thesis: Stanford University. 2013.

[37] A survey of fault tolerance mechanisms and checkpoint/restart implementations for high performance computing systems / Ifeanyi P Egwutuoha, David Levy, Bran Selic [и др.] // The Journal of Supercomputing. 2013. Т. 65, № 3. С. 1302–1326.

[38] Meyer Hugo, Rexachs Dolores, Luque Emilio. RADIC: A FaultTolerant Middleware with Automatic Management of Spare Nodes* // Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA) / The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp). 2012. С. 1.

[39] Adaptive load balancing for MPI programs / Milind Bhandarkar, Laxmikant V Kalé, Eric de Sturler [и др.] // Computational Science-ICCS 2001. Springer, 2001. С. 108–117.

[40] More scalability, less pain: A simple programming model and its implementation for extreme computing / Ewing L Lusk, Steven C Pieper, Ralph M Butler [и др.] // SciDAC Review. 2010. Т. 17, № 1. С. 30–37.

[41] Anderson J Chris, Lehnardt Jan, Slater Noah. CouchDB: The definitive guide. O’Reilly Media, Inc., 2010.

[42] Lakshman Avinash, Malik Prashant. Cassandra: A decentralized structured storage system // ACM SIGOPS Operating Systems Review. 2010. Т. 44, № 2. С. 35–40.

[43] Dean Jeffrey, Ghemawat Sanjay. MapReduce: Simplified data processing on large clusters // Communications of the ACM. 2008. Т. 51, № 1. С. 107–113.

[44] Apache Hadoop YARN: Yet Another Resource Negotiator / Vinod Kumar Vavilapalli, Arun C Murthy, Chris Douglas [и др.] // Proceedings of the 4th annual Symposium on Cloud Computing / ACM. 2013. С. 5.

[45] Armstrong Joe. Making reliable distributed systems in the presence of software errors. Ph.D. thesis: The Royal Institute of Technology Stockholm, Sweden. 2003.

[46] Degtyarev A., Gankevich I. Evaluation of hydrodynamic pressures for autoregression model of irregular waves // Proceedings of 11th International Conference “Stability of Ships and Ocean Vehicles”, Athens. 2012. C. 841–852.

[47] Degtyarev A., Gankevich I. Wave Surface Generation Using OpenCL, OpenMP and MPI // Proceedings of 8th International Conference “Computer Science & Information Technologies”. 2011. C. 248–251.

[48] Degtyarev A.B., Reed A.M. Modelling of Incident Waves Near the Ship’s Hull (Application of autoregressive approach in problems of simulation of rough seas) // Proceedings of the 12th International Ship Stability Workshop. 2011.

[49] Degtyarev A.B., Reed A.M. Synoptic and Short-Term Modeling of Ocean Waves // Proceedings of 29th Symposium on Naval Hydrodynamics. 2012.

[50] Fischer Michael J, Lynch Nancy A, Paterson Michael S. Impossibility of distributed consensus with one faulty process // Journal of the ACM (JACM). 1985. Т. 32, № 2. С. 374–382.

[51] The impossibility of implementing reliable communication in the face of crashes / Alan Fekete, Nancy Lynch, Yishay Mansour [и др.] // Journal of the ACM (JACM). 1993. Т. 40, № 5. С. 1087–1107.

Список опубликованных по теме ВКР работ

[1] Degtyarev A., Gankevich I. Hydrodynamic Pressure Computation under Real Sea Surface on Basis of Autoregressive Model of Irregular Waves // Physics of Particles and Nuclei Letters. 2015. Vol. 12, no. 3. P. 389–391. URL: http://dx.doi.org/10.1134/S1547477115030073.

[2] Degtyarev A., Gankevich I. Calculation Scheme for Wave Pressures with Autoregression Method // 14th International Ship Stability Workshop. 2014. P. 135–139.

[3] Дегтярев А. Б., Ганкевич И. Г. Вычисление гидродинамических давлений под реальной морской поверхностью на основе авторегрессионной модели нерегулярного волнения // Труды XLV НТК “Проблемы мореходных качеств судов, корабельной гидромеханики и освоения шельфа” (Крыловские чтения). 2013. С. 25–29.

[4] Degtyarev A., Gankevich I. Hydrodynamic pressure computation under real sea surface on basis of autoregressive model of irregular waves. 2013.

[5] Degtyarev A., Gankevich I. Evaluation of hydrodynamic pressures for autoregression model of irregular waves // Proceedings of 11th International Conference “Stability of Ships and Ocean Vehicles”. Athens, Greece: 2012. P. 841–852.

[6] Degtyarev A., Gankevich I. Efficiency Comparison of Wave Surface Generation Using OpenCL, OpenMP and MPI // Proceedings of 8th International Conference “Computer Science & Information Technologies”. Yerevan, Armenia: 2011. P. 248–251.

[7] Ганкевич И. Г., Дегтярев А. Б., Соэ Моэ Лвин. Сравнение эффективности применения MPI и OpenCL для генерации волновой поверхности // Морские интеллектуальные технологии. 2010. Т. 4. С. 10–13.

[8] Gankevich I., Degtyarev A. Efficient processing and classification of wave energy spectrum data with a distributed pipeline // Computer Research and Modeling. 2015. Vol. 7, no. 3. P. 517–520. URL: http://crm-en.ics.org.ru/journal/article/2301/.

[9] Gankevich I., Tipikin Y., Gaiduchok V. Subordination: Cluster management without distributed consensus // International Conference on High Performance Computing Simulation (HPCS). 2015. P. 639–642. Outstanding poster paper award.

[10] Novel Approaches for Distributing Workload on Commodity Computer Systems / I. Gankevich, Y. Tipikin, A. Degtyarev et al. // Computational Science and Its Applications – ICCSA 2015 / Ed. by O. Gervasi, B. Murgante, S. Misra et al. Springer International Publishing, 2015. Vol. 9158 of *Lecture Notes in Computer Science*. P. 259–271. URL: http://dx.doi.org/10.1007/978-3-319-21410-8_20.

[11] Ганкевич И. Г., Дегтярев А. Б. Методы распределения нагрузки на многопроцессорную систему // Процессы управления и устойчивость. 2014. Т. 1, № 17. С. 295–300.

[12] Gankevich I., Degtyarev A. Model of distributed computations in virtual testbed // Proceedings of IX International Conference on Computer Science and Information Tecnologies (CSIT'13). Yerevan, Armenia: 2013. P. 240–244.

[13] Virtual Supercomputer as basis of Scientific Computing / A. Bogdanov, A. Degtyarev, V. Korkhov et al. // Horizons in Computer Science Research / Ed. by T. S. Clary. Nova Science Publishers, 2015. Vol. 11. P. 159–198.

[14] Running applications on a hybrid cluster / A. Bogdanov, I. Gankevich, G. V. et al. // Computer Research and Modeling. 2015. Vol. 7, no. 3. P. 475–483. URL: http://crm-en.ics.org.ru/journal/article/2295/.

[15] Applications of on-demand virtual clusters to high performance computing / I. Gankevich, S. Balyan, S. Abrahamyan et al. // Computer Research and Mod-

eling. 2015. Vol. 7, no. 3. P. 511–516. URL: http://crm-en.ics.org.ru/journal/article/2300/.

[16] Profiling Scheduler for Efficient Resource Utilization / A. Bogdanov, V. Gaiduchok, N. Ahmed et al. // Computational Science and Its Applications – ICCSA 2015 / Ed. by O. Gervasi, B. Murgante, S. Misra et al. Springer International Publishing, 2015. Vol. 9158 of *Lecture Notes in Computer Science*. P. 299–310. URL: http://dx.doi.org/10.1007/978-3-319-21410-8_23.

[17] Constructing Virtual Private Supercomputer Using Virtualization and Cloud Technologies / I. Gankevich, V. Korkhov, S. Balyan et al. // Lecture Notes in Computer Science. 2014. Vol. 8584. P. 341–354.

[18] Virtual private supercomputer: Design and evaluation / I. Gankevich, V. Gaiduchok, D. Gushchanskiy et al. // IEEE conference publication. Vol. 6710358. 2013. P. 1–6.

A. Верификация авторегрессионной модели

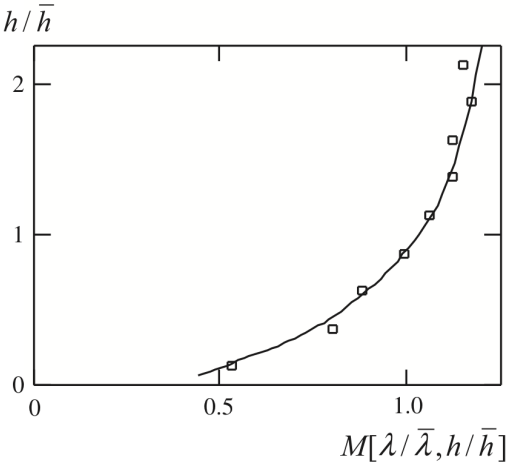

Рис. 13: Регрессия длин волн на их высоты. □ – модель, сплошная линия – эксперимент.

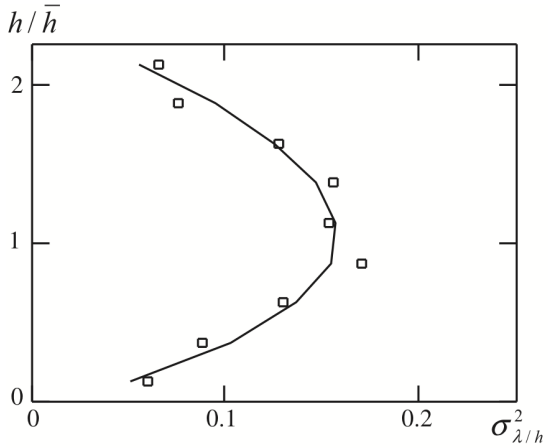

Рис. 14: Скедастическая кривая длин и высот волн. □ – модель, сплошная линия – эксперимент.

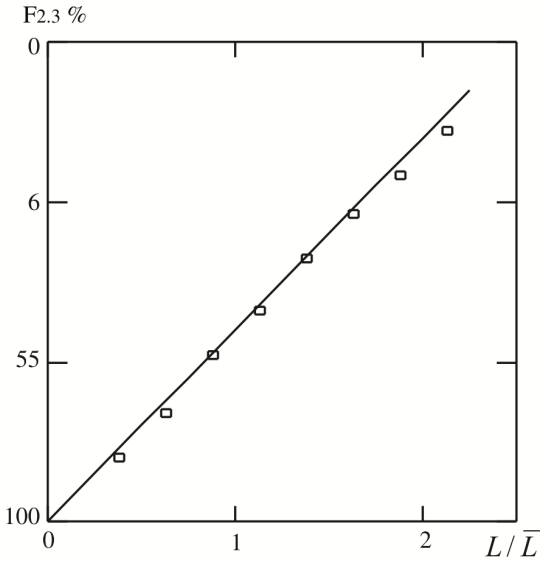

Рис. 15: Распределение высот волн. □ – модель, сплошная линия – эксперимент.

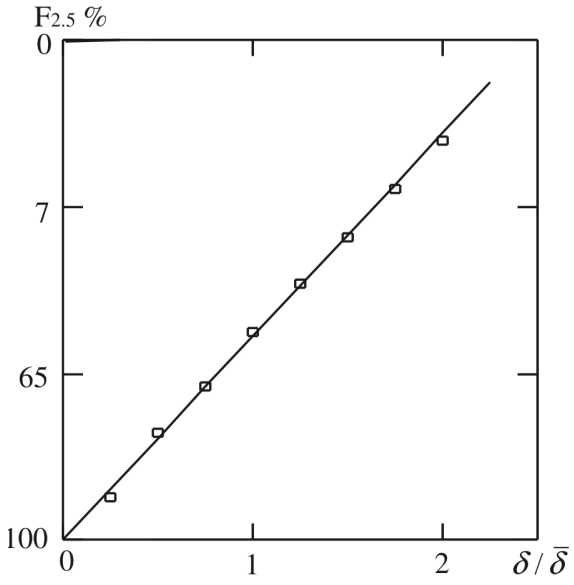

Рис. 16: Распределение уклонов волн. □ – модель, сплошная линия – эксперимент.

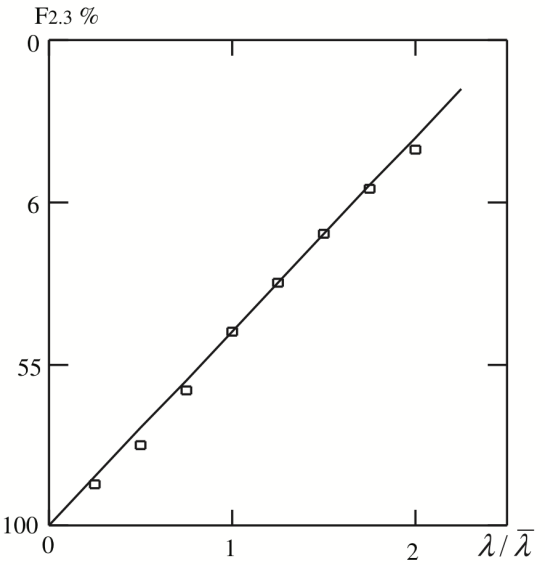

Рис. 17: Распределение длин волн. □ – модель, сплошная линия – эксперимент.

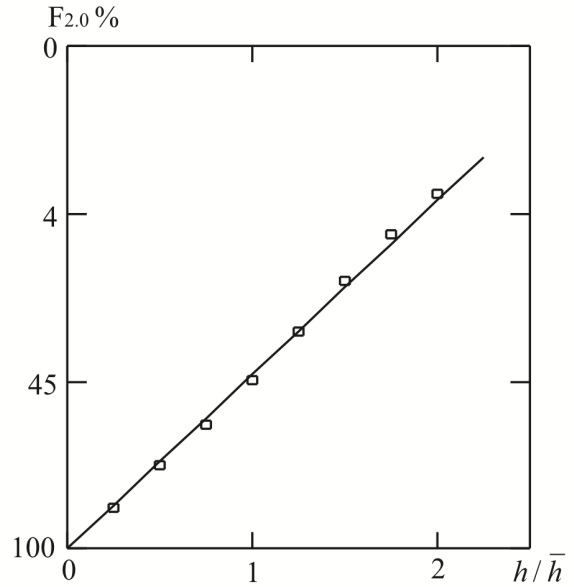

Рис. 18: Распределение высрт волн. □ – модель, сплошная линия – эксперимент.

Благодарности

Исследования были проведены с использованием вычислительных ресурсов Ресурсного Центра «Вычислительный центр СПбГУ» (http://cc.spbu.ru/) при поддержке грантами РФФИ №16-07-01111, №16-07-00886 и грантом СПбГУ №0.37.155.2014.